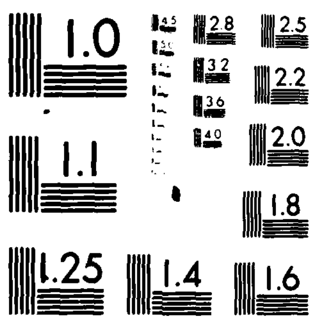
AD-A112 458   DUKE UNIV DURHAM NC DEPT OF MECHANICAL ENGINEERING A--ETC   F/G 6/19
DEVELOPMENT OF A STRAIN RATE DEPENDENT LONG BONE INJURY CRITERI--ETC(U)
JAN 82   T K HIGHT   AFOSR-81-0062

UNCLASSIFIED   AFOSR-TR-82-0138   NL

1 OF 2
AD
A112458

A124



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

AFOSR-TR- 82-0138

(11)

AD A112458

DEVELOPMENT OF A STRAIN RATE DEPENDENT LONG BONE INJURY CRITERION FOR USE WITH THE ATB MODEL

FINAL REPORT

CONTRACT AFOSR-81-0062

Timothy K. Hight
Assistant Professor
Dept. of Mechanical Engineering
and Materials Science
Duke University
Durham, N.C. 27706

January 12, 1982

82 03 22

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFOSR-TR- 82 -0138 / 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER: AD-A112 458
4. TITLE (and Subtitle): Development of a Strain Rate Dependent Long Bone Injury Criterion for Use with the ATB Model.	5. TYPE OF REPORT & PERIOD COVERED: Final Report
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Timothy K. Hight	8. CONTRACT OR GRANT NUMBER(s): AFOSR-81-0062
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Dept. of Mechanical Engineering & Materials Sci. Duke University Durham, N.C. 27706	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F 2312/D9
11. CONTROLLING OFFICE NAME AND ADDRESS: AFOSR/NL Bolling Air Force Base Washington, D.C. 20332	12. REPORT DATE: January 12, 1982
	13. NUMBER OF PAGES: 102
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release;
distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by bl… …)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See Over

JAN 20 1982

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

ABSTRACT

An improved version of a previously reported long bone injury criterion has been developed. This new criterion was based on fitting the Ramberg-Osgood equation, through optimization techniques, to existing stress-strain-strain rate curves for compact bone and using this equation to solve for the strain at a time step given the current stress and the strain from the previous time. Refinements have also been made to the geometric modeling of the bones where adequate data existed. When coupled with the (ATB) model and relations between ultimate stress and strain rate, this criterion has provided an effective means of judging long bone injury potential.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

I. INTRODUCTION

The Articulated Total Body (ATB) Model has been developed as a tool to predict and analyze the response of the human body to potential injury causing environments. This model has been primarily used for the analysis of vehicular crashes, and has been extremely useful in predicting gross motion of the body. The model also predicts the loads on each of the articulated joints. The USAF has been involved in the development and testing of this computer model and has applied it to the analysis of the response of pilots to ejection from jet aircraft. During these events the body is subjected to high accelerations from the ejection itself, to impacts of the limbs with each other and with the cockpit, and to "wind flail" from the high velocity wind stream which impacts the body after clearing the aircraft. It is hoped that a better understanding of these events will lead to safer designs and lower injury rates.

Despite the excellent results obtained from the ATB model, the current version does not provide the information necessary to predict whether the event will or will not cause an injury. It is therefore necessary to augment the current model by the addition of some sort of injury criterion in order for the full potential of the ATB to be realized. With this addition, the model can be used to compare different acceleration profiles, restraint systems and other variables as to their injury preventing potential. Currently these assessments must be made, in a very qualitative way, on the basis of motion of the limbs and loads in the joints.

Since the majority of serious injuries resulting from ejections are bone fractures, it is of particular interest to estimate the likelihood of long bone fracture. (It should be noted that a separate computer model, the Head-Spine Model,

is specifically designed to study the question of head and spine injuries, and we have consciously ignored this aspect of injury).

The research carried out under this minigrant was begun by the principal investigator during a Summer Faculty Research Program stay at WPAFB. The major objectives of that research program were to establish a simplified injury criterion for bone and to implement that criterion in a computer program (BREAK) compatible with the existing ATB. Due to the constraints of such a short time period (10 weeks), the project was necessarily restricted to a rather narrow scope. Within these constraints the objectives were accomplished. A detailed report of this project can be found in the Final Report submitted to SCEEE, and published as a technical report by AFAMRL (1), but the significant developments will be summarized here.

Injury criteria reported in the literature have focused on two distinct approaches- "experimental" and "analytical". The experimental approach comes largely from the automobile industry and involves the experimental determination of loads which, under simulated crash tests, cause injury to cadavers. For long bones this is restricted to the Femur Injury Criterion which is a measure of the axial load applied to the shaft of the femur which causes fracture. The current Federal Standard is 1700 lb force without any loading rate dependence. A number of authors have suggested different criteria with loading rate sensitivity (2,3). Even so, the basic fault of these criteria is their specificity - they say nothing about other long bones or other loading conditions. This type of criterion was therefore inappropriate to a study of ejection seat injuries.

The alternate type of injury criterion is based on the analytic approach using the material and geometric properties of the bone and the calculated stresses. This approach has the advantage of being applicable to any loading situation if the properties of the bone are known. This was the approach taken.

The material properties of bone are highly variable, depend on a large number of parameters and are time dependent (4,5,6). In order to accurately model the behavior of bone, consistant data including complete strain rate dependent stress strain curves are needed for human bone. The data which most nearly fulfilled these criteria was the work of McElhaney (7) which was for embalmed human bone in compression. His results indicated the following relation between ultimate stress (σ) and strain rate ($\overset{\circ}{\varepsilon}$)

$$\sigma = 4200 \text{ Log } \overset{\circ}{\varepsilon} + 33000. \tag{1}$$

This equation, after modifications, became the basis for the injury criteria developed. Two major changes to this equation were made - the strain rate dependency was replaced by a stress rate dependence, and the constant term was modified to bring the results in line with published static results for fresh human compact bone. The rate modification was accomplished by first finding a relation between elastic modulus and strain rate, and then differentiating the elastic stress strain relation with respect to time for a constant strain rate. The second modification was based on $\overset{\circ}{\varepsilon} = .001$ as a static strain rate and was straightforward. The resulting equation for compression (and in psi units) is

$$\sigma = 3936 \log \overset{\circ}{\varepsilon} + 12000 \tag{2}$$

Similar expressions were generated for tension and shear based on the assumption that the material behavior (rate dependence) would be the same in the other modes and that only the constant offset would change. These equations formed the basis of the injury criterion models developed that summer.

An alternative to these criterion was generated using stress pulse duration, rather than stress rate, as the time variable. This alternate approach was examined because most published criteria from the auto industry are based on pulse duration.

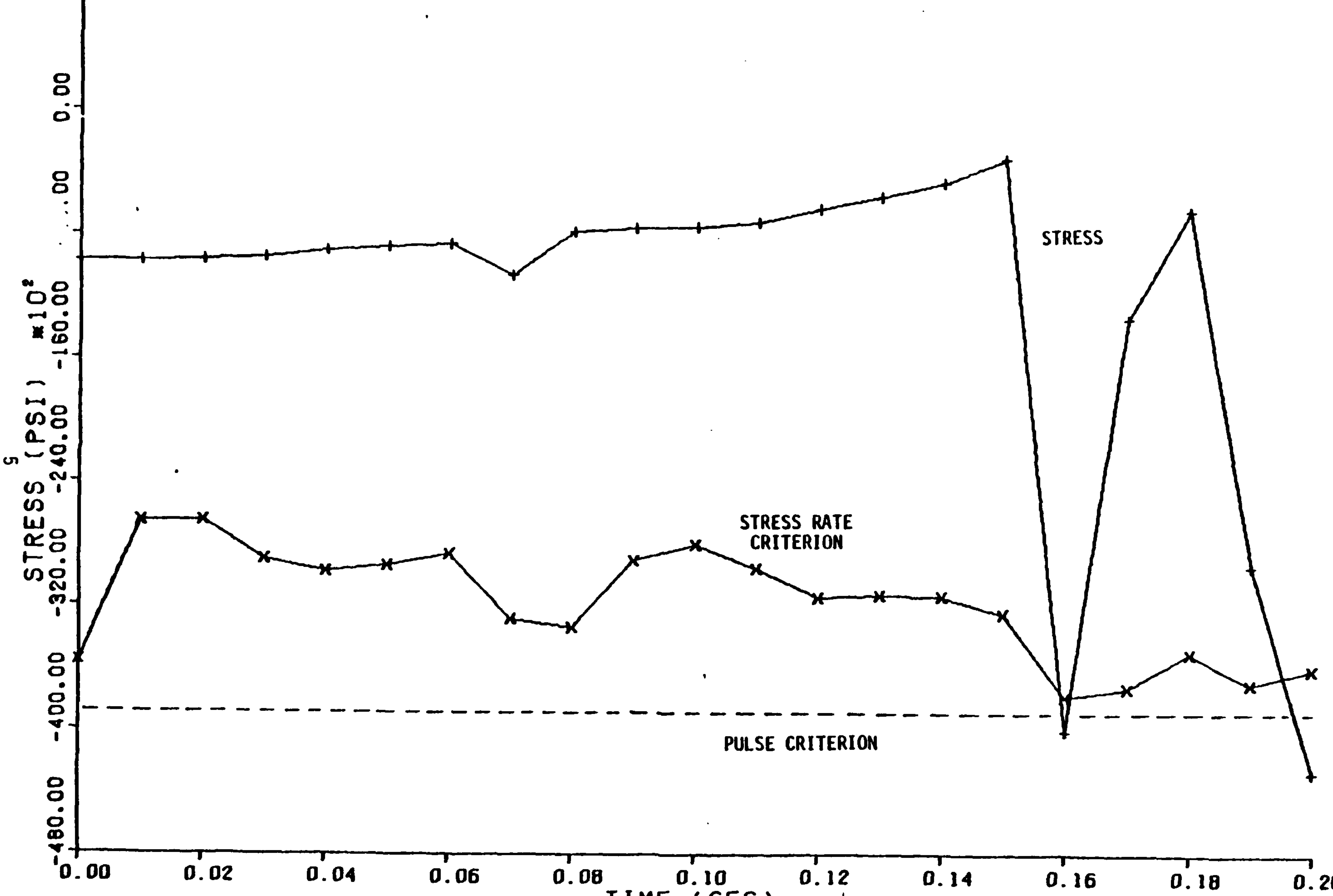
The pulse duration was approximated by assuming a sinusoidal pulse for the stress. The "pulse" criterion tended to be slightly less conservative than the stress rate criteria.

The stress calculations were based on a very simplified geometry, namely straight, uniform, isotropic hollow tubes for all the long bones. Principal stresses were then calculated at some predetermined number of sections equally spaced along the bone axis and the peak tension, compression and shear stresses were monitored. The peak stresses as a function of time for a particular limb were therefore not necessarily stresses at the same point, the position along the axis of the bone could vary with time also.

A test case was run based on an ejection simulation to demonstrate the program. Results are presented in the Technical Report mentioned above and a sample figure is reproduced here. Figure 1 shows the maximum compressive stress as a function of time for the left lower arm. There was contact at .07 secs and some high level, higher frequency loading at the elbow near the end of the data used (only the first 200 msec of the event were examined). The stress rate dependence of the allowable stress is evident as is the slightly less conservative nature of the pulse criterion. Note that the pulse criterion has only been calculated for the maximum stress pulse and hence appears as a constant.

This summarizes the state of this project at the start of the minigrant.

COMPRESS. STRESS VS ALLOWABLE



LLA RESPONSE TO ATB EJECTION SIMULATION (7/24/80)

Figure 1

II. OBJECTIVES

As outlined in the grant proposal, there were five main objectives of this investigation, and they were:

1. Improve the efficiency of the developed program, BREAK.

2. Search out and reexamine the available data base on bone material properties.

3. Develop more precise constituitive equations for bone.

4. Provide a statistical analysis of the available data.

5. Redefine the injury criteria based on the new information.

III. RESULTS

In this section each of the objectives described in the previous section will be addressed in turn, and the outcome of each investigation will be detailed.

A. The greatest problem with BREAK, as discussed in the proposal, has been the lack of compatibility between the output of the ATB Model and the input needs of BREAK. Basically the problems are ones of transformation of coordinate systems and reconstruction of contact forces from resultants. As an example, suppose that a limb contacts a panel resulting in normal and friction forces. The output of the ATB Model will include the linear and angular position, velocity and acceleration of the center of mass of the segment along with the forces and moments resulting at the joints of the segment. In addition, the resultant contact force and the point of contact will be given. For BREAK, since it calculates stress levels at a number of points along the axis of the bone, all forces must be known in local coordinates and the contact forces must be known in a vector sense, rather than in magnitude only. The reconstruction of the contact forces is a laborious task which repeats the work already done by the ATB program. In fact, all necessary information exists within the ATB program at some point during the calculations and, as proposed earlier, a file could be set up containing the appropriate information. (See program CONTACT in Appendix). Unfortunately, this has not yet been carried out. The major difficulty has been the lack of available memory on the CDC-CYBER computer system used by AFAMRL. All "non essential" data are currently eliminated during processing of the program in order to save space, and that loss currently includes the data which is needed for BREAK. Therefore the current version of BREAK (see CONTACT in Appendix) includes all of the manipulations which must be performed to transform the given data into the needed format.

Another important change to the program dealt with bone geometry. Rather than continue using uniform, hollow tubes to represent bones, we have tried to imitate some of the variations in actual bone geometry. Using the limited amount of available bone geometry data (8,9) we have been able to model variations in cross-section properties along the bone axes (for 20 to 80% of distal distance) for the femur and tibia. No data was found for the upper extremity bones. For this model the bone is assumed to be made up of a discontinuous series of short sections of uniform hollow tubes.

It was necessary to scale the available data, which appeared to be from approximately 50th percentile subjects, up to a 95th percentile man so that the bone data would coincide with the other pilot data. This was accomplished by making the assumption that, for different size bodies, the stress levels in the bones would be relatively constant. Using the body weight and limb length, relative ratios of cross-sectional area and area moment of inertia could be generated using axial compressive stress and bending stress equations. The data used, in both original and scaled form, are shown in Figures 2 and 3. See program STRESS for a complete description of the analysis, along with the calculation of stresses at various cross-sections.

B. It has been well established that the elastic and ultimate properties of compact bone are loading rate dependent (e.g. 7,10). Ultimate stress levels typically vary by a factor of two or more over a range of 5 or 6 decades of strain rates. This effect is simply too large to ignore. However, the use of ultimate stress values alone, even if they are well known, is insufficient.

The ATB model represents the body as a series of rigid, though resilient, links connected by springs. As such, for each segment the end loads and motions are well

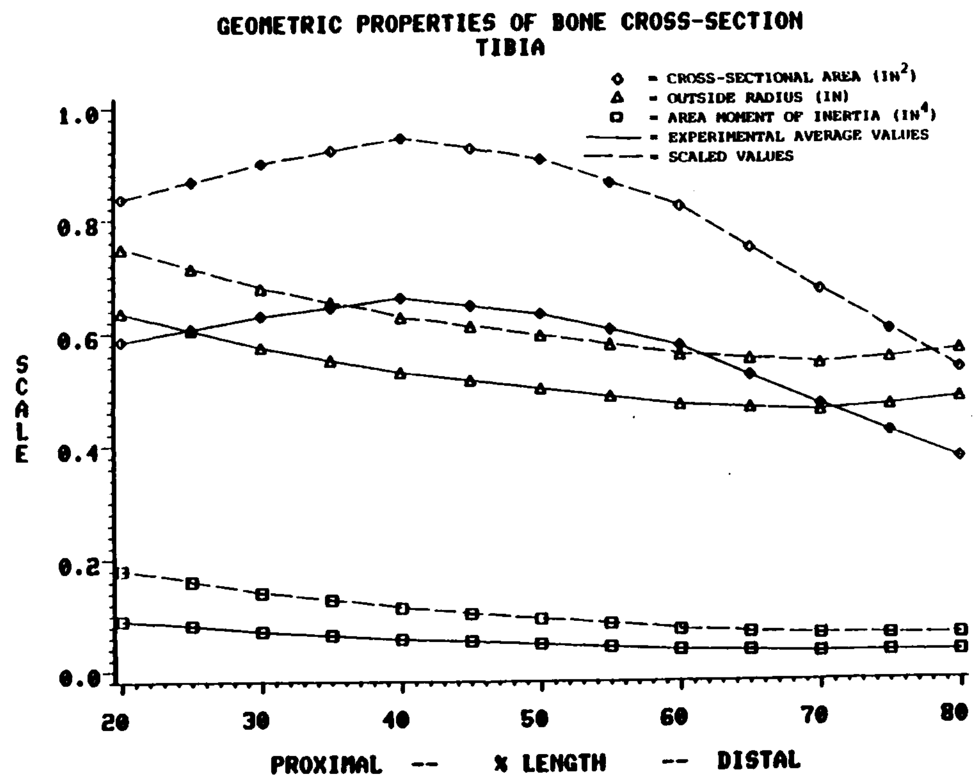

GEOMETRIC PROPERTIES OF BONE CROSS-SECTION
TIBIA
◇ = CROSS-SECTIONAL AREA (IN^2)
△ = OUTSIDE RADIUS (IN)
□ = AREA MOMENT OF INERTIA (IN^4)
—— = EXPERIMENTAL AVERAGE VALUES
– – = SCALED VALUES
SCALE
1.0
0.8
0.6
0.4
0.2
0.0
20
30
40
50
60
70
80
PROXIMAL -- % LENGTH -- DISTAL

FIGURE 2

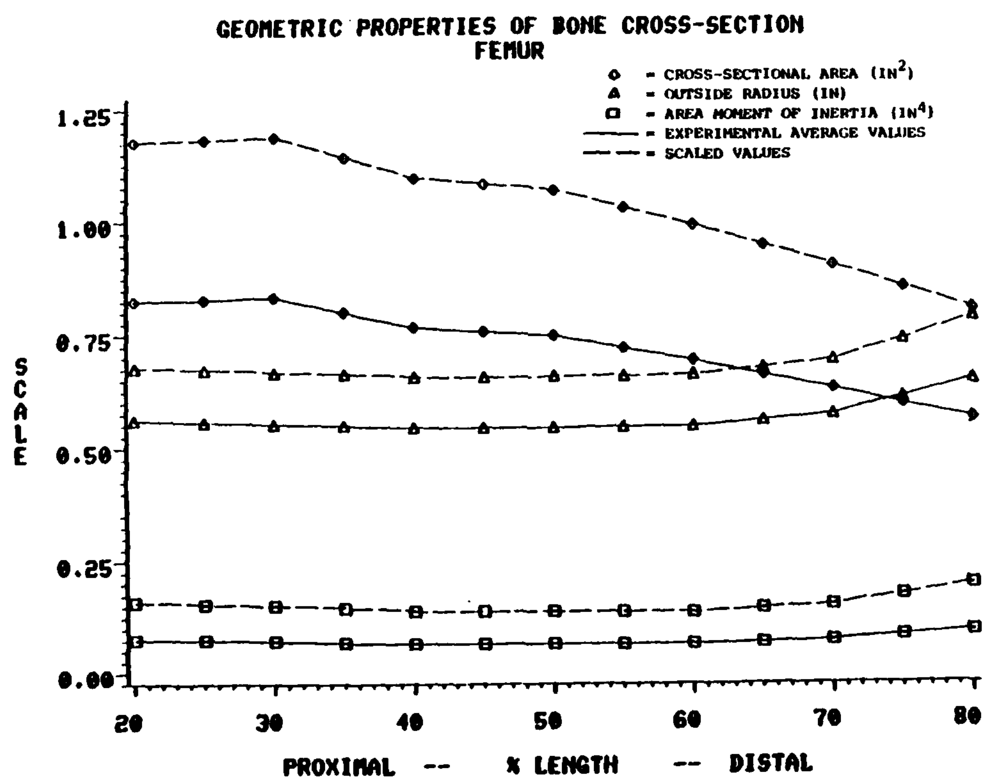

GEOMETRIC PROPERTIES OF BONE CROSS-SECTION
FEMUR
= CROSS-SECTIONAL AREA (IN^2)
= OUTSIDE RADIUS (IN)
= AREA MOMENT OF INERTIA (IN^4)
= EXPERIMENTAL AVERAGE VALUES
= SCALED VALUES
SCALE
1.25
1.00
0.75
0.50
0.25
0.00
20
30
40
50
60
70
80
PROXIMAL -- % LENGTH -- DISTAL

FIGURE 3

defined. These knowns, along with the bone geometries, are sufficient to describe the stress state at any timestep and at any point along the limb. The strains are then related to the stress through the theory of elasticity (for an elastic structure). For bone, the modulus of elasticity is a function of strain rate, and that function is not analytically invertable.

Three sets of stress-strain curves for various strain rates have been uncovered in the literature. McElhaney published the results of both bovine and embalmed human bone in compression using constant strain rates between 10^{-3} and 1500 sec^{-1} (7). More recently, Wood has reported on the tensile properties of fresh cranial bone for strain rates from $.003 \leq \dot{\varepsilon} \leq 150\ sec^{-1}$ (11). This data is of little utility for the ATB model because Wood used very small bone samples and because cranial bone is significantly different in structure from diaphysial compact bone. Crowninshield and Pope (12) have also published results for compact bovine bone in tension over strain rates of $.00167 \leq \dot{\varepsilon} \leq 250$. Their results indicate a much larger plastic strain region for longitudinal samples than do the results previously reported. However, Burstein *et.al*. (13) have also shown considerable plastic strain in bovine bone in tension and very little in compression. In Burstein's tests strain rates were not reported, but loading duration was between 1/10 and 1/2 sec.

The above three papers constitute a very small data set and none of the results are for fresh human long bone. Other researchers have dealt with portions of this problem, however they usually report only ultimate properties and not the full stress-strain curves. Lewis and Goldsmith (14), for example, used a split Hopkinson Bar method to measure the fracture stress of bovine bone in compression. The strain histories were pulses rather than constant strain rate. The fracture strain rates were calculated by dividing the fracture strain by the time and these rates varied

over approximately 6 orders of magnitude. These results are consistently higher than the results of McElhaney. Wright and Hayes (15) reported ultimate tensile strength for fresh bovine bone for $5.3 \times 10^{-4} \leq \dot{\varepsilon} \leq 237 \text{ sec}^{-1}$. They also present two characteristic load-displacement curves, but no stress-strain data.

C. Much progress has been made in the area of establishing analytic functions which describe the relationship between stress, strain and strain rate. Using available published data and our optimizing curve fit computer program we have been able to get good fits of the data using a standard Ramberg-Osgood equation,

$$\varepsilon = \frac{\sigma}{c\dot{\varepsilon}^{d}} + a\sigma^{N}\dot{\varepsilon}^{b} \quad . \tag{3}$$

This equation is a standard equation for modeling visco elastic-plastic behavior of materials (16). Results for the three available sets of curves are shown numerically in Table 1, and graphically in Figures 4, 5, and 6. The ability of this function to fit such a wide variation of results is encouraging.

Because of the orders of magnitude variations in the strain rate, certain constants in equation (3) were extremely sensitive. For example, the constant multiplier "a" for the plastic term varied by as much as 55 orders of magnitude from author to author. It was therefore necessary to take our optimization approach (see program MYFIT in Appendix) to minimize the function F (the sum of weighted errors of data-point fits), where F is defined by equation (4).

$$F = \sum_{i=1}^{M} [H(\varepsilon_p - \varepsilon_o)]^N + r \sum_{i=1}^{M} \langle gi \rangle^z \tag{4}$$

BONE TYPE	Test Mode	c	d	a	N	b
emb. human femur	C	3551	.0671	6.12×10^{-13}	6.53	-.3740
bovine (long.)	T	1694	.0180	3.71×10^{-68}	45.3	-2.336
cranial compact	T	2200	.0567	3.68×10^{-12}	7.66	-.4127

Table 1 (for stress in ksi)

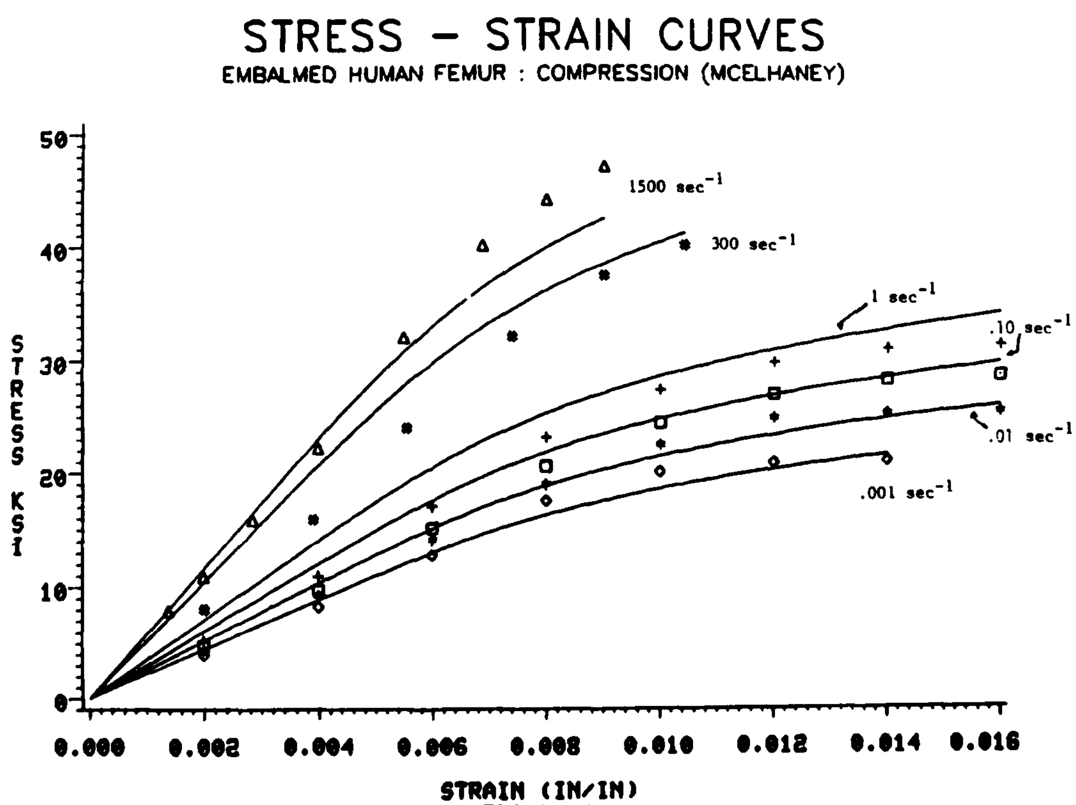


FIGURE 4

STRESS - STRAIN CURVES
BOVINE : TENSION (CROWNINSHIELD & POPE)
LONGITUDINAL

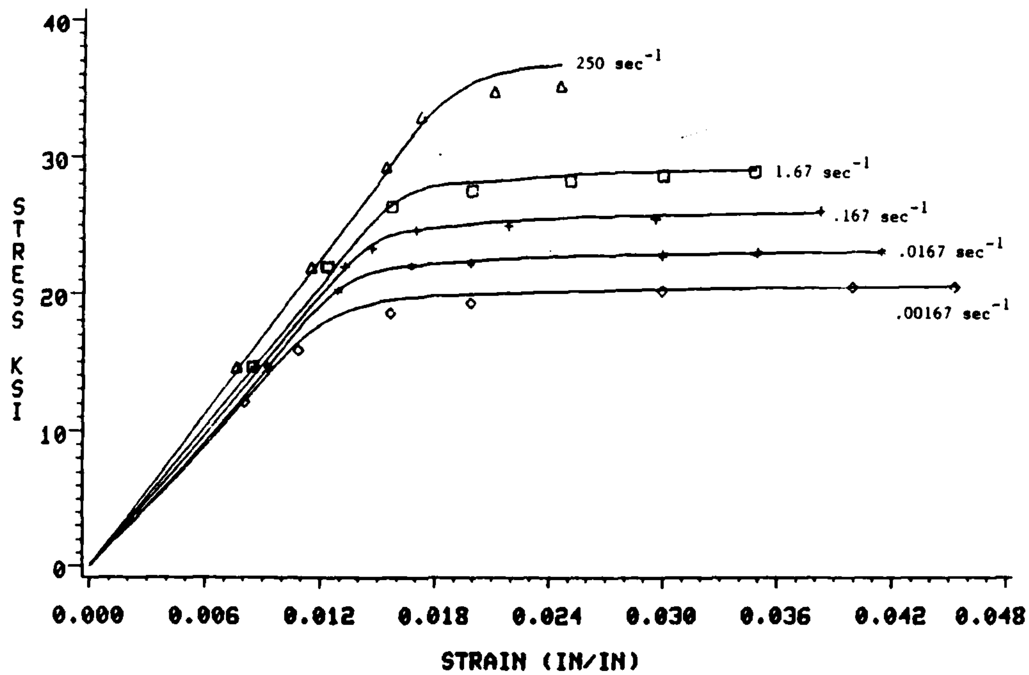


FIGURE 5

STRESS - STRAIN CURVES
CRANIAL COMPACT BONE : TENSION (WOOD)

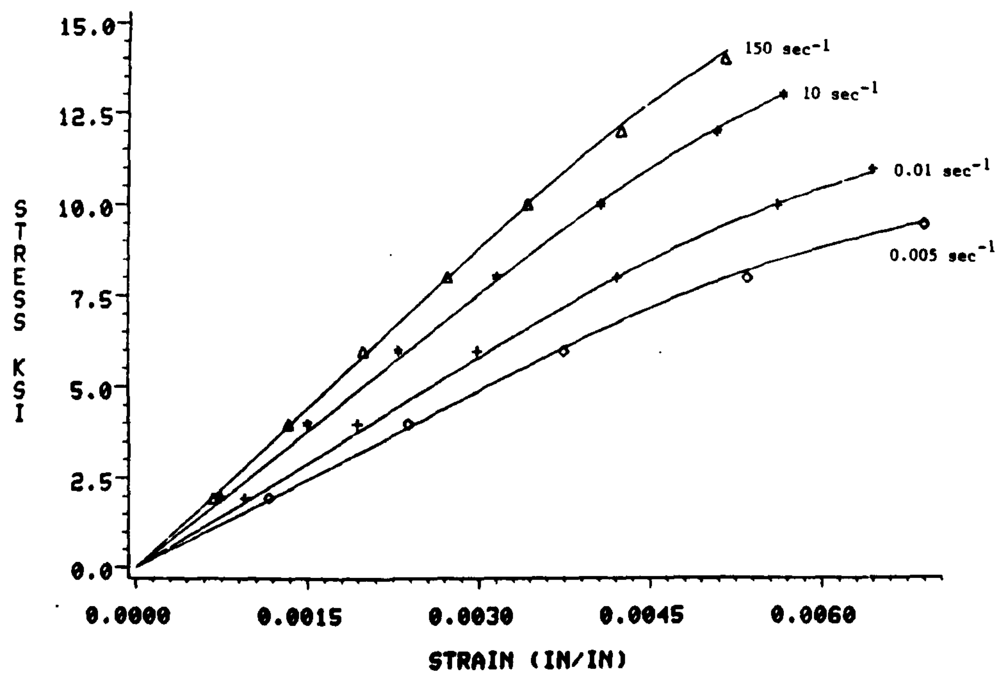


FIGURE 6

and where

M = the number of experimentally observed points of $(\sigma,\varepsilon,\dot{\varepsilon})$

ε_p = strain predicted by curve

ε_o = strain observed by experiment

r,H = constant multipliers for weighting

N,z = constant powers for weighting

$\langle gi \rangle$ = exterior penalty function for weighting

The objective function F is minimized in each of the five coordinate directions (a,b,N,d.c) using a combination of golden section search and parabolic interpolation methods (17). The resultant coordinates are the coefficients of the equation that best fits the set of observed data.

The results of the optimization process are the curves shown in Figures 4, 5 and 6. It can be seen that there is excellent agreement between most of the predicted and experimental results. The largest variations occur in the very high strain rate tests of McElhaney. Forcing the function to try and fit the 1500 sec^{-1} data tends to also skew the rest of the curves. As an experiment, the 1500 sec^{-1} data was removed from the sample population and the curves refit to the remaining data. This trial showed a very good correlation with the remaining data and a high prediction at 1500 sec^{-1}. As this is the only data available at this high a strain rate it is difficult to draw any conclusions about this response.

D. It is obvious from the previous section that no meaningful statistical analysis can be performed on the meager amount of available data. However, it is interesting to compare the various results and this is shown in Figures 7, 8, 9 and 10. These figures show predicted stress-strain curves at a given strain rate based on various authors' results. Be aware first of all that these curves, except

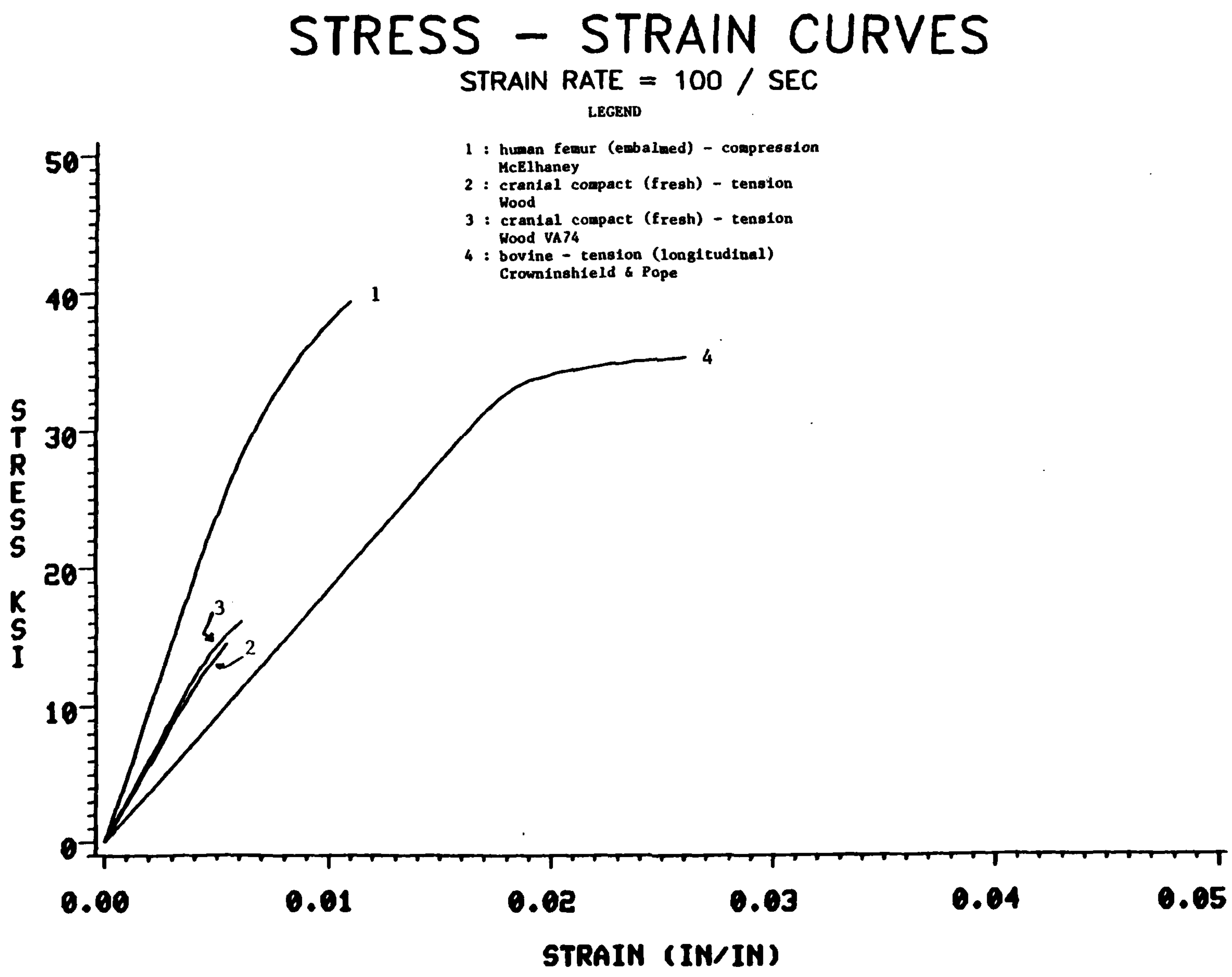

STRESS - STRAIN CURVES
STRAIN RATE = 100 / SEC
LEGEND
1 : human femur (embalmed) - compression
McElhaney
2 : cranial compact (fresh) - tension
Wood
3 : cranial compact (fresh) - tension
Wood VA74
4 : bovine - tension (longitudinal)
Crowninshield & Pope
1
2
3
4
STRESS KSI
0
10
20
30
40
50
0.00
0.01
0.02
0.03
0.04
0.05
STRAIN (IN/IN)

FIGURE 7

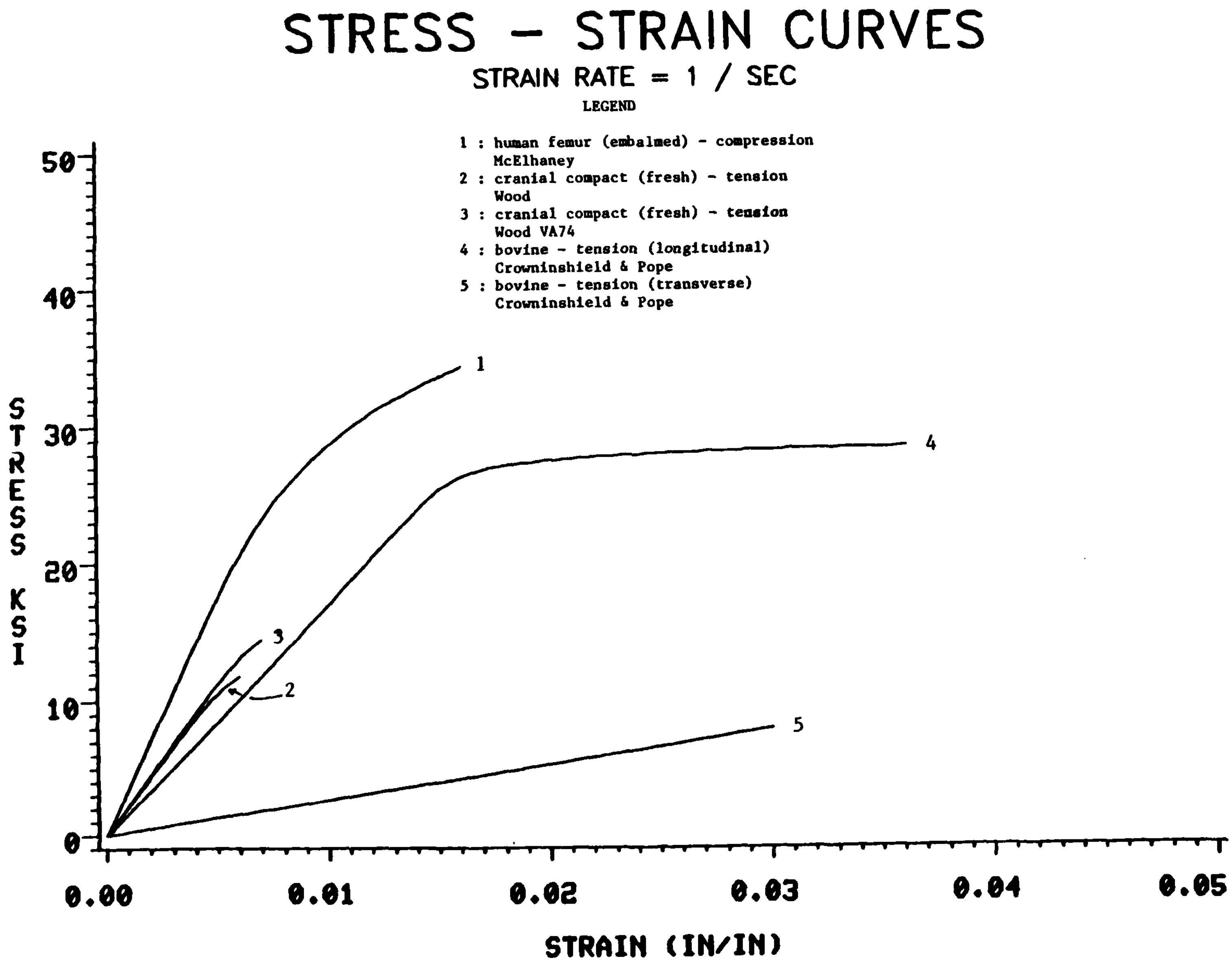

STRESS - STRAIN CURVES
STRAIN RATE = 1 / SEC
LEGEND
1 : human femur (embalmed) - compression McElhaney
2 : cranial compact (fresh) - tension Wood
3 : cranial compact (fresh) - tension Wood VA74
4 : bovine - tension (longitudinal) Crowninshield & Pope
5 : bovine - tension (transverse) Crowninshield & Pope
STRESS KSI
0
10
20
30
40
50
0.00
0.01
0.02
0.03
0.04
0.05
STRAIN (IN/IN)
1
2
3
4
5

FIGURE 8

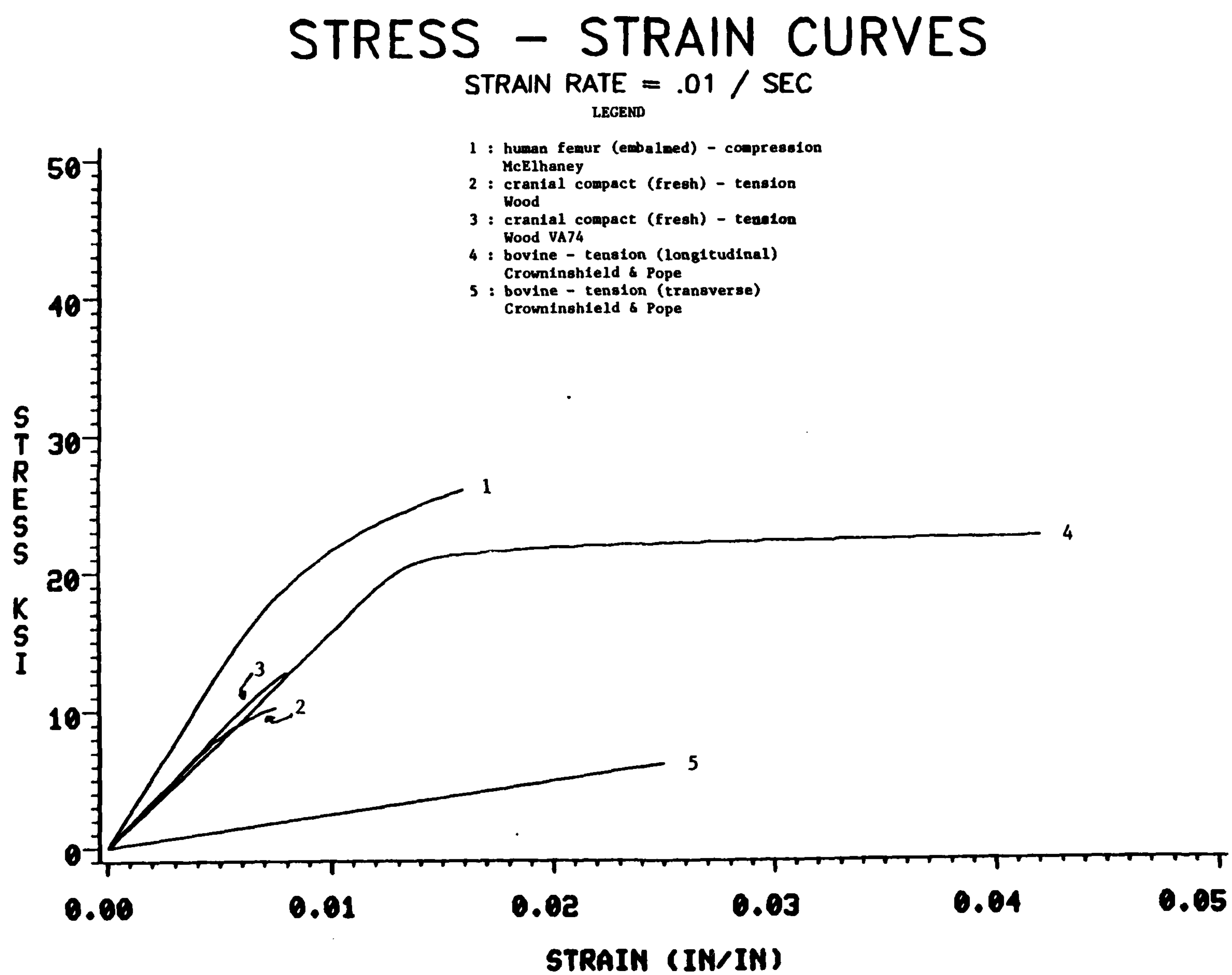

STRESS - STRAIN CURVES
STRAIN RATE = .01 / SEC
LEGEND
1 : human femur (embalmed) - compression McElhaney
2 : cranial compact (fresh) - tension Wood
3 : cranial compact (fresh) - tension Wood VA74
4 : bovine - tension (longitudinal) Crowninshield & Pope
5 : bovine - tension (transverse) Crowninshield & Pope
STRESS KSI
0
10
20
30
40
50
0.00
0.01
0.02
0.03
0.04
0.05
STRAIN (IN/IN)
1
2
3
4
5

FIGURE 9

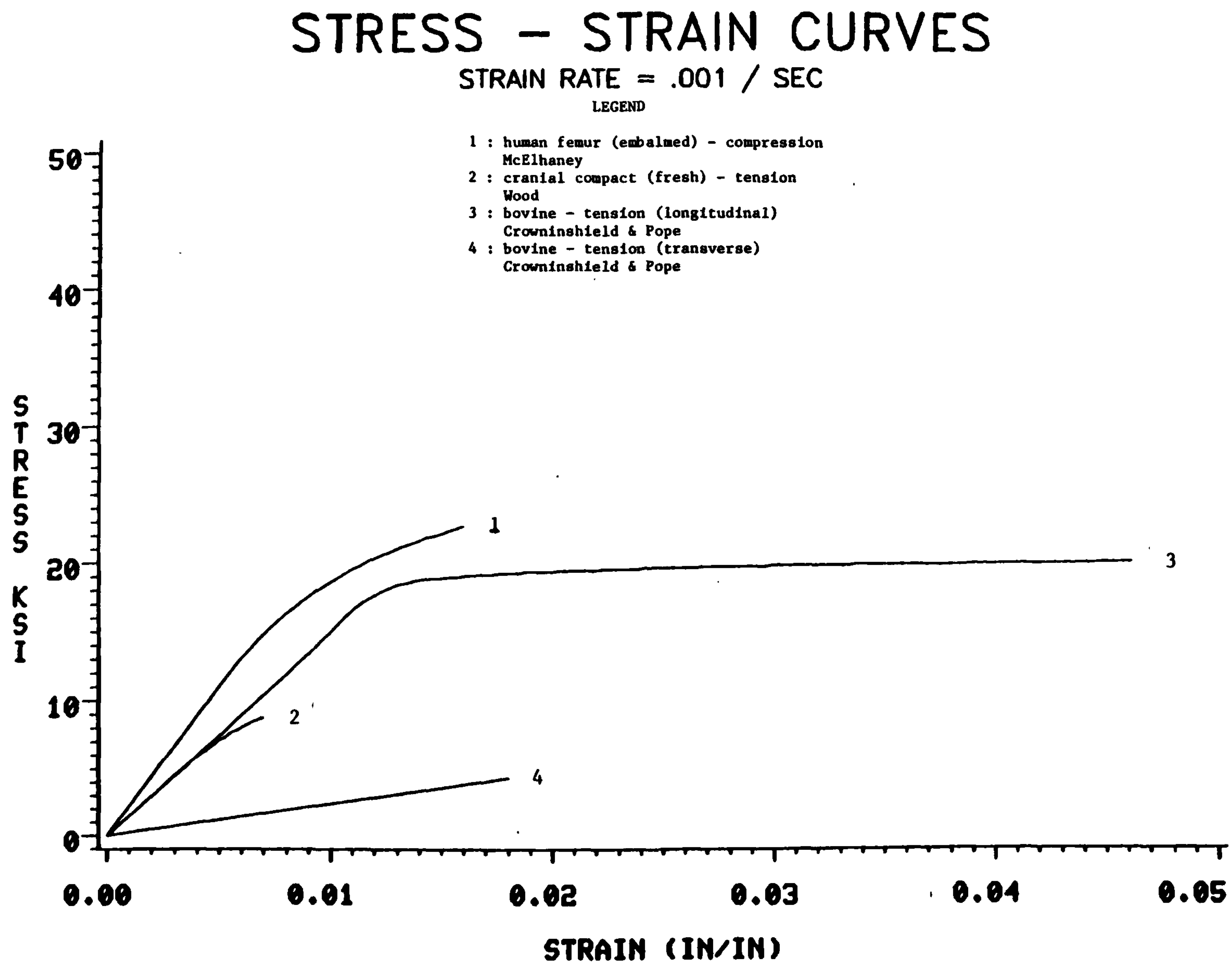

STRESS - STRAIN CURVES
STRAIN RATE = .001 / SEC
LEGEND
1 : human femur (embalmed) - compression
McElhaney
2 : cranial compact (fresh) - tension
Wood
3 : bovine - tension (longitudinal)
Crowninshield & Pope
4 : bovine - tension (transverse)
Crowninshield & Pope
STRESS KSI
0
10
20
30
40
50
0.00
0.01
0.02
0.03
0.04
0.05
STRAIN (IN/IN)
1
2
3
4

FIGURE 10

for the two by Wood, all represent different tests-embalmed human femur in compression; human cranial compact bone in tension; and bovine femur in tension for longitudinal and transverse directions. In that sense we are comparing apples to oranges. For this reason, these figures are shown only for general comparison. It had been hoped that a more direct comparison of similar data from the literature could have been made.

The cranial bone samples are generally weaker with stiffnesses bracketed by the compression results (stiffest and highest strength) and the tension results (less stiff and most plastic). Results for transverse tension samples are shown to be significantly lower in strength and stiffness. As mentioned earlier, the only significant plastic zone was found by Crowninshield and Pope for bovine bone in longitudinal tension. The results of Wood indicate that there is some consistency among data from tests on similar material and under similar conditions.

A study was carried out during the initial summer's work to assemble a small subset of this data, namely the ultimate strengths for fresh human compact bone from femurs tested statically in the longitudinal direction. The results are reproduced in Table 2. For the four modes of failure, the means and standard deviations are, respectively; tension 99.3/26/3, compression 174.4/34.2, shear 69.0/14.8 and bending 161.4/11.9, all *10**6 N/m**2. These figures illustrate the wide discrepencies which exist for even a narrowly defined subset of available data. The wide variations in these values could be due to a number of causes including the age and moisture content of the specimens and the test procedures utilized by the investigators.

E. The injury criteria, as first developed, were based on relationships between the ultimate stress and the strain rate (equation 1). This equation was then modified, based on a constant strain rate approximation, to yield a relation between ultimate

Table 2

Ultimate Strengths of Fresh Human Compact Bone From Femurs, Tested Statically and Stressed in the Longitudinal Direction
($x10^6$ N/m^2/$x10^3$ psi)

Tension	Compression	Shear	Bending
122/17.7*	159/23.1*	53.1/7.7*	152/22.0&
86.5/12.5*	193/28.0$	82.4/11.9*	153/22.1¢
133/19.3$	134.5/19.5¢	71.6/10.4$	157/22.8*
76.2/11.0+	210.9/30.6+		164/23.8*
78.9/11.4+			181/26.2*

Compiled from various sources reported in (*) Reilly and Burstein (6), (+) Evans (4), ($) Reilly and Burstein (18), (&) Mather (19) and (¢) Vose and Kubala (20).

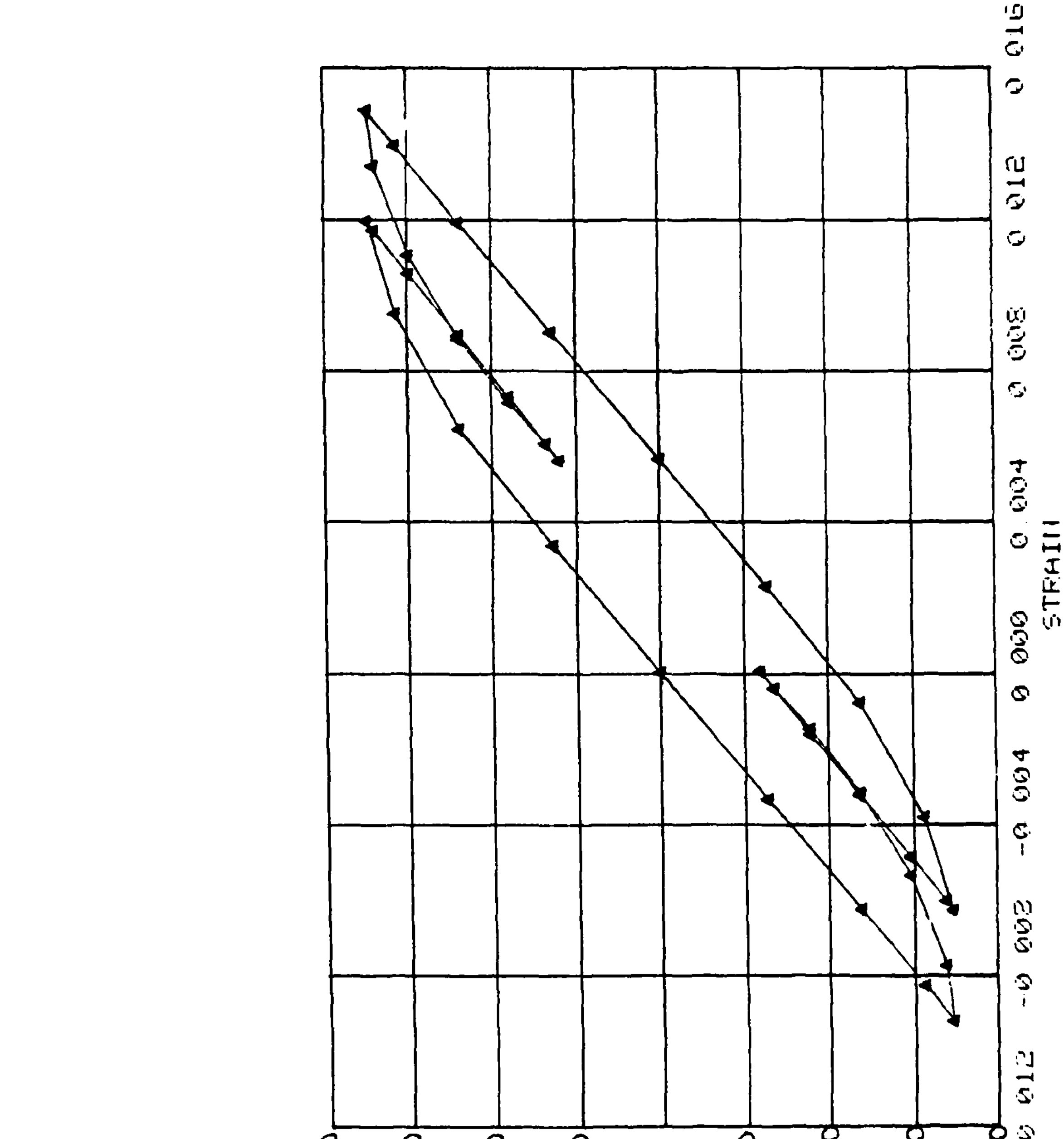
stress and stress rate which could be calculated directly from the available stress time histories (equation 2). This solution is rather simplistic and not entirely satisfactory. A more systematic approach based on the Ramberg-Osgood equation (equation 3) has yielded more promising results.

The elemental question is, how to calculate strain when only the stress time history is known and the relation between stress and strain is strain rate dependent? It is clear on inspection that equation (3) does not invert or separate into convenient parts, particularly since the factors b and d are, in general, not integers. The approach taken has been one of a numerical approximation based on the following equation at time step i:

$$\varepsilon_i = \frac{\sigma_i}{c} \frac{1}{\left(\frac{\varepsilon_i - \varepsilon_{i-1}}{\Delta t}\right)^d} + a\sigma_i^N \left(\frac{\varepsilon_i - \varepsilon_{i-1}}{\Delta t}\right)^b \qquad (5)$$

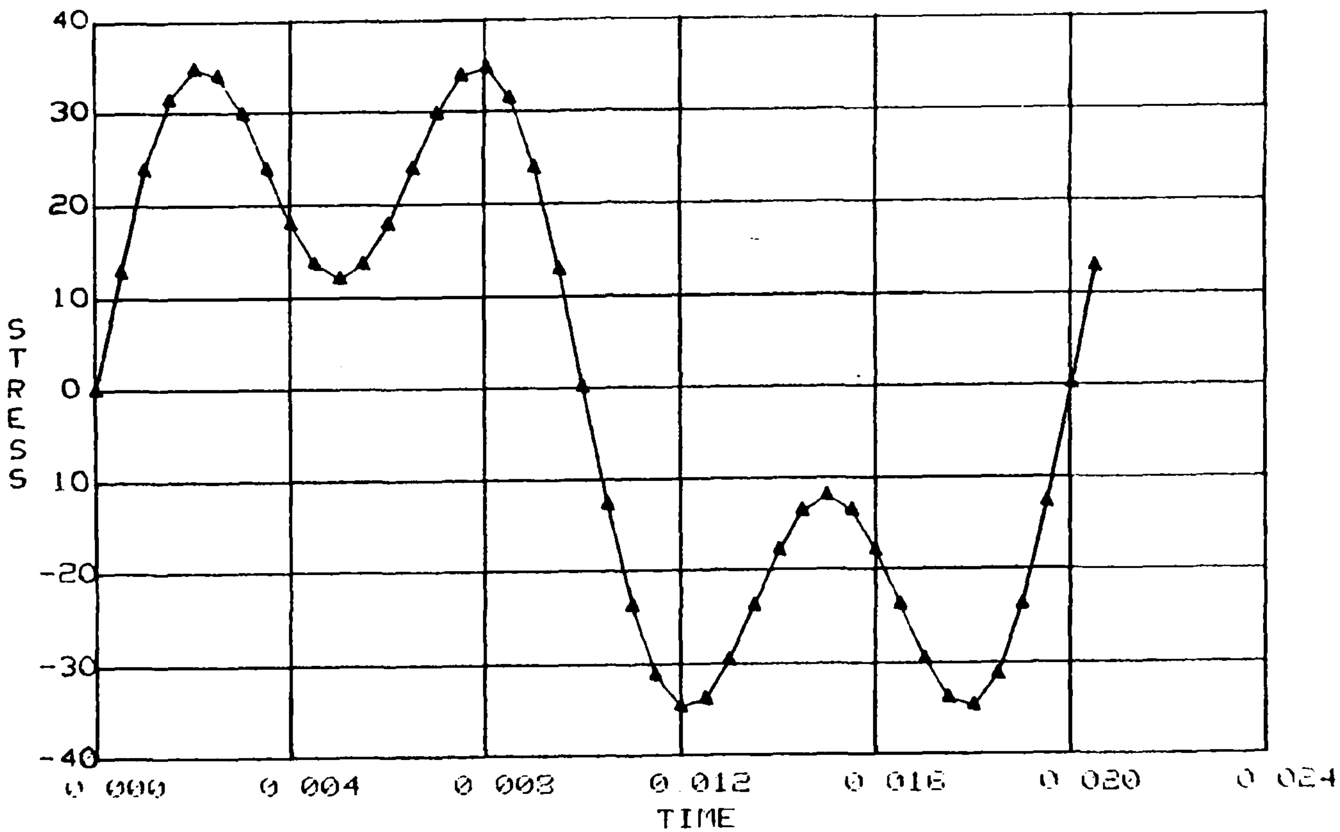
Given the stress at a time step and the strain at the previous time step, equation (5) can be solved numerically for the current strain. This solution procedure is easily initialized by assuming zero stress and strain at time = 0. Solution of this equation has been successfully accomplished and Figure 11 shows an example stress strain response to the sinusoidal stress wave shwon in Figure 12. The solution displays the expected hysteresis type response.

There were a number of stumbling blocks which had to be overcome in order to achieve this solution. The second term of equation (5), which accounts for the plastic portion of the response, can have a severe destabilizing effect on the solution procedure, depending on the values of the constants and the strain rate. It was therefore necessary to use a two step method based on the Regula-Falsi technique. An initial solution was generated using only the first term of the equation to be used as a starting point. With this initial guess as a basis the final solution



STRESS STRAIN RESPONSE

FIGURE 11



STRESS TIME HISTORY

FIGURE 12

was obtained using both terms. This two step procedure greatly reduced the incidence of non convergence, but, even using double precision, did not entirely eliminate these difficulties if the stress levels were too high.

A second problem was encountered due to the relaxation phase of the loading. Since b and d are not integers, a negative strain rate raised to such a power has no meaning. Furthermore, it can be assumed from standard hysteresis loop response that relaxation is essentially an elastic phenomenon. For these reasons, only the first term of the equation, with appropriate absolute values, is used during relaxation.

The final problem arose from the plastic deformation which may occur. Since equation (5) presumes a zero strain at zero stress, an appropriate plastic strain offset must be added to the solution after any relaxation. This plastic effect is clearly shown in Figure 9.

Using the solutions generated by equation (5) it is now possible to follow the strain and strain rate response of the system and to directly use equation (1) to predict ultimate stress. This procedure enables one to realistically use available ultimate stress values as a fracture criterion.

The point may be made that yield rather than ultimate stress would be a more conservative, and more meaningful injury criterion. Yield stress would unquestionably be more conservative, though some questions must first be answered. First, of course, yield point must be defined. This definition is arbitrary, depending on the material. For most engineering metals, yield is defined in terms of that stress which produces a plastic strain of .002. There is no such accepted standard for bone and all figures reported in the literature are for ultimate strengths. If a yield point definition could be agreed upon for bone, then yield strengths could be defined from the solution of equation (5).

A perhaps more interesting question is, what is the biological effect in living bone of exceeding the "yield strength?" No known investigation has been made into this phenomenon and its relation to injury. In fact, the overall level of understanding of bone fracture mechanics is quite low. For example, it is conceivable that reconstruction in living bone would overcome yielding effects without injury.

IV. CONCLUSIONS AND RECOMMENDATIONS

Given the available bone fracture data and the constraints of the current ATB model, a reasonable mechanism has been developed for utilizing a long bone fracture injury criterion for the ATB model. Specifically, this mechanism is based on a mathematical representation of the behavior of bone using the Ramberg-Osgood equation and on an approximation of the stress-time history of the bone generated using the ATB joint and contact forces. The specific injury criterion which should be employed is still under consideration. Options include: ultimate strength, which would be non conservative but for which considerable data exist; yield strength, which would be conservative, but for which a definition, and hence data, do not exist; some fraction of either of these as a factor of safety; or some other criterion such as maximum strain or strain energy. At present the ultimate strength, reduced by a factor of safety, is thought to be the best criterion. (The pulse criterion discussed in the first report (1) is felt to have value only as a comparison to existing automobile standards and hence to have no applicability to the pilot ejection problem.) The resulting injury criterion program remains in the "post processor" mode, i.e. it is not integrated into the ATB model, and this is the preferred configuration until the methodology becomes more fixed.

The Ramberg-Osgood equation appears to provide a very good means of mathematically modelling the stress-strain-strain rate behavior of bone. The fact that this equation, through proper optimization techniques, can be used to represent the three very different responses of bone shown here from the literature leads one to suspect that, whatever the "true" response of bone may be, it can be adequately described by equation (3). This flexibility, coupled with the dearth of good data on dynamic bone properties, was a prime motivating factor in taking this approach.

Further areas of inquiry point strongly in the direction of establishing a more comprehensive and consistent data base. The state-of-the-art in modeling has clearly outstripped both the available information on bone material properties and whole bone geometry, and the understanding of the mechanisms of bone fracture. A thorough understanding of these properties and mechanisms is necessary before any strides can be made toward formulating a more effective long bone injury criterion. The present criterion is designed to adapt to new information as it becomes available through modification of the constants in the Ramberg-Osgood equation and represents, in its present form, the most sophisticated injury criterion available.

BIBLIOGRAPHY

1. Hight, T., "Long bone injury criteria for use with the Articulated Total Body Model", AFAMRL-TR-81-3, 1981.

2. King, J.J., W.R.S. Fan and R.J. Vargovick, "Femur load injury criteria - a realistic approach," 17th Stapp Conference, paper #730984, pp. 509-525, 1973.

3. Viano, D.C., "Considerations for a Femur Injury Criterion," 21st Stapp Conference, October 19-21, New Orleans, paper #77095, pp. 443-373, 1977.

4. Evans, F.G., Stress and Strain in Bone, Charles C. Thomas, Springfield, IL. 1957.

5. Curry, J.D., "The mechanical properties of bone," Clin. Orth. and Rel. Res. Vol. 73, Nov-Dec. 1970, pp. 210-231.

6. Reilly, D.T. and A.H. Burstein, "The mechanical properties of cortical bone," JBJS, Vol. 56A No. 5, July 1974, pp. 1001-1022.

7. McElhaney, J.H., "Dynamic response of bone and muscle tissue," J. Appl. Physiology, Vol. 21, No. 4, pp. 1231-1236, 1966.

8. Minns, R.J., Bremble, G.K. and Campbell, J. "The geometrical properties of the human tibia," Tech. Note, J. Biomech. Vo. 8, pp. 253-255, 1975.

9. Piziali, R.L., Hight, T.K. and Nagel, D.A., "Geometric properties of human leg bones," J. Biomech., V. 13, pp. 881-885, 1980.

10. Panjabi, M.M., White, A.A. and Southwick, W.D., "Mechanical properties of bone as a function of rate of deformation," JBJS V. 55A, pp. 322-330, March 1973.

11. Wood, J.L., "Dynamic response of human cranial bone," J. Biomech. V. 4, pp. 1-12, 1971.

12. Crowninshield, R.D. and M.H. Pope, "The response of compact bone in tension at various strain rates," Annal. Biomed. Eng. V. 2, pp. 217-225, 1974.

13. Burstein, A.H., Currey, I.D., Frankel, V.H. and Reilly, D.T., "The ultimate properties of bone tissue: the effects of yielding," J Biomech., V. 5, pp. 34-44, 1972.

14. Lewis, J.L. and Goldsmith, W., "The dynamic fracture and prefracture response of compact bone by split Hopkinson bar methods," J. Biomech. V. 8, No. 1, pp. 27-40, 1975.

15. Wright, T.M. and Hayes, W.C., "Tensile testing of bone over a wide range of strain rates: effects of strain rate, microstructure and density," Med. Biol. Eng. V. 14, No. 6, pp. 671-680, 1976.

16. McLellan, D.L., "Constitutive equations for mechanical properties of structural materials," AIAAJ, V. 5, No. 3, pp. 446-450, 1967.

17. Forsythe, G.E., Computer Methods for Mathematical Computations, Prentice-Hall, New Jersey, pp. 182-190, 1977.

18. Reilly, D.T. and A.H. Burstein, "The elastic and ultimate properties of compact bone tissue," J. Biomech., V. 8, pp. 393-405, 1975.

19. Mather, B.S., "The effect of variation in specific gravity and ash content on the mechanical properties of human compact bone," J. Biomech., V. 1, pp. 207-210, 1968.

20. Vose, G.P. and A.L. Kubala, "Bone strength - its relationship to x-ray determined ash content," Human Biol., V. 31, pp. 261-270, 1959.

PERSONNEL:

Principal Investigator: Timothy K. Hight
Department of Mechanical Engineering
and Materials Science
Duke University
Durham, N.C. 27705

Graduate Student: John F. Brandeau

(While no support for this graduate student was received from this grant, he did contribute much of the work reported here while working toward his M.S. degree. A thesis is currently in preparation _in absentia_ with an expected completion date of May 1982).

PUBLICATIONS:

in preparation, "Mathematical modeling of the stress-strain-strain rate behavior of bone using the Ramberg-Osgood equation," T.K. Hight, J.F. Brandeau (probable journal - J. Biomech.).

in preparation, "Development of a strain rate dependent injury criterion for long bones," T.K. Hight, J.F. Brandeau (probable journal - J. Biomech.).

INTERACTIONS:

Seminar presented at WPAFB/AFAMRL August 7, 1981 "Improved long bone injury criteria for use with the ATB model."

APPENDIX

```
:*****************************************************************************
:                                                                            *
: PROGRAM NAME : MYFIT                                                       *
: WRITTEN BY :  J.F. BRANDEAU                                                *
: COMPILER(S) : WATFIV  (DOUBLE PRECISION)                                   *
:       ----------------------------------------------------                 *
: PURPOSE :  FIT A FUNCTION OF N VARIABLES TO A SET OF M OBSERVED DATA       *
: POINTS BY MINIMIZING THE DIFFERENCES BETWEEN OBSERVED AND PREDICTED        *
: VALUES.  THIS IS DONE USING SUBROUTINE FMIN (A COMBINATION OF GOLDEN       *
: SEARCH & PAROBOLIC INTERPOLATION METHODS) IN EACH OF THE COORDINATE        *
: DIRECTIONS (X).  THIS PROCEDURE STOPS WHEN ONE OR MORE TERMINATION          *
: CRITERIA ARE MET :                                                         *
:    1)  ALL STEPS THROUGH TWO CONSECUTIVE SERIES ARE ABSOLUTELY LESS        *
:        THAN TOL.                                                           *
:    2)  FRACTIONAL CHANGE OF FUNCTION VALUE IS ABSOLUTELY LESS THAN         *
:        TOL THROUGH ONE SERIES.                                             *
:    3)  MAXIMUM # OF SERIES IS EXCEEDED.                                    *
:       ----------------------------------------------------                 *
: PROGRAM VARIABLES : SUGGESTED VALUES TO START WITH IN ()                   *
:                                                                            *
: NSCALE : CONTROLS LENGTH OF INTERVAL SENT TO FMIN.  THE VALUE OF X(I)      *
:  IS MULTIPLIED BY (1+NSCALE(I)) AND (1-NSCALE(I)).  IF THE INTERVAL        *
:  INCLUDES ZERO, THIS ALLOWS FOR MAJOR CHANGES OF VARIABLE X(I) IN THE      *
:  ABSOLUTELY SMALLER DIRECTION.  NOTE --  IF NSCALE(I) IS 0.0, THE          *
:  VALUE OF X(I) DOES NOT CHANGE IN THE PROGRAM.  IN THIS WAY THE VAR-       *
:  IABLE X(I) CAN BE FIXED. (1.0 FOR ALL).                                   *
:                                                                            *
: TOL : CONVERGENCE LIMIT FOR ALL CRITERIA (1.0D-7).                         *
:                                                                            *
: MAXFN : MAXIMUM # OF FUNCTION EVALUATIONS ALLOWED (25).                    *
:                                                                            *
: TIMES : MULTIPLIER FOR EACH ABSOLUTE ERROR (VALUE WHICH WILL MAKE THE      *
:  ERROR GREATER THAN 1.0).                                                  *
:                                                                            *
: UP : POWER WHICH (TIMES * ABS. ERROR) IS RAISED TO (2.0).                  *
:       ----------------------------------------------------                 *
: PENALTY CONTROLS : PENALIZES FUNCTION IF ABS ERROR AT ANY POINT IS         *
:  GREATER THAN A SPECIFIED AMOUNT.                                          *
:                                                                            *
: ERROR : ALLOWABLE ERROR WITHOUT PENALTY (1.5D-3).                          *
:                                                                            *
: Z : POWER WHICH (ABS. ERROR * TIMES) IS RAISED TO.  THIS IS THE            *
:  PENALTY STEP FUNCTION (IF ABS. ERROR GT. ERROR ; PENALTY = ABS.           *
:  ERROR * TIMES ) ** Z, --  ELSE PENALTY = 0)  THIS IS A RUNNING SUM.       *
:  (Z IS USUALLY 2.0).                                                       *
:       ----------------------------------------------------                 *
: OTHERS VARIABLES :                                                         *
:                                                                            *
: X & COEFF : MANTISSA & EXPONENT OF VARIABLES.  AFTER EACH SERIES X IS      *
:  NORMALIZED TO BETWEEN 1.0 & 10.0 TO ALLOW LARGE DELTA X'S AND AID         *
```

```
C  CONVERGENCE CRITERIA ACCURACY.  THE PRODUCT X(I) * COEFF(I) IS THE  *
C  VALUE OF THE VARIABLE COEFFICIENT.                                  *
C                                                                      *
C STRESS : OBSERVED VALUES OF STRESS (KSI).                            *
C                                                                      *
C RATE : OBSERVED VALUES OF STRAIN RATE (1/SEC).                       *
C                                                                      *
C Y : OBSERVED VALUES OF STRAIN (IN/IN).                               *
C                                                                      *
C YAPROX : PREDICTED VALUES OF STRAIN (IN/IN).                         *
C            -------------------------------------------------         *
C CCOMP OPTIONS : FORM C$OPTIONS CCOMP=?????                           *
C                                                                      *
C 3 : OUTPUTS FUNCTION AND POINT OF EVALUATION FOR EACH EVALUATION.    *
C 4 : OUTPUT VALUE OF X & FMIN INTERVAL (AX & BX), PENALTIES AND       *
C     FUNCTION VALUES FOR EACH COORDINATE DIRECTION.                   *
C 5 : ALLOWS READING FROM INPUT LIST AT END OF PRORAM FOR X & COEFF.   *
C     PROGRAM ALWAYS READS THESE FROM FILE #5 REGARDLESS.  FILE #1     *
C     VALUES WILL OVERRIDE THESE.                                      *
C 6 : OUTPUTS PENALTY SUMMATION AT TERMINATION.                        *
C 7 : OUTPUTS LIST OF OBSERVED AND PREDICTED VALUES AT TERMINATION.    *
C            ----------------------------------------------            *
C I/O REQUIREMENTS :                                                   *
C                                                                      *
C FILE #1 : CONTAINS INPUT VARIABLES AS DESCRIBED ABOVE FOLLOWING $DATA*
C           CARD AT END OF PROGRAM.                                    *
C FILE #4 : CONTAINS OBSERVED VALUES.  THE FIRST RECORD MUST BE A TITLE*
C FILE #5 : CONTAINS X & COEFF ON TWO RECORDS.  UPDATED AT TERMINATION.*
C                                                                      *
C***********************************************************************
C$OPTIONS CCOMP=0
      ON ERROR GO TO 70
      IMPLICIT REAL * 8 (A-H, O-Z)
      REAL NSCALE(5)
      DIMENSION X(5), STRESS(60), Y(60), RATE(60), YAPROX(60)
      DIMENSION A(5), DIFF(60)
      INTEGER TITLE(20)
      COMMON STRESS, Y, RATE, YAPROX, M, SSQ, DIFF, TIMES
      COMMON /WEIGHT/ R, Z, ERROR, NUMPEN, PENAL
      COMMON /SEARCH/ EPS
      COMMON /FUN1/ COEFF(5), TOTAL, UP, A, I
      EXTERNAL FUNCT
      DATA NSCALE /0.90, 0.90, 1.00, 1.00, 1.00 /
C
C SET PROGRAM PARAMETERS
C
      Z = 2.0D0    ;    UP = 2.0D0    ;    R = 10.0D0
      TOL = 1.0D-7  ;  ERROR = 1.0D-4  ;    MAXFN = 20
      TIMES = 1.0D3
C
```

```
C
C CALCULATE SQUARE ROOT OF MACHINE EPSILON FOR SUBROUTINE FMIN
C
      EPS = 1.0D00
   10 EPS = EPS / 2.0D0
      TOL1 = 1.0D0 + EPS
      IF (TOL1 .GT. 1.0D00) GO TO 10
      EPS = DSQRT(EPS)
      KOUNT = NUMPEN = 0
      READ (1,*) N
      READ (5,*) (X(I), I = 1, N)
      READ (5,*) (COEFF(I), I = 1, N)
C5    READ (1,*) (X(I), I = 1, N)
C5    READ (1,*) (COEFF(I), I = 1, N)
      READ (4,95) TITLE
      M = 1
   20 READ (4,*,END=25) RATE(M), Y(M), STRESS(M)
      M = M + 1
      GO TO 20
   25 M = M - 1
      I = 1 ; POINT = X(I)
      WRITE (3,98)  TITLE
      WRITE (3,105) TOL, UP, R, Z, ERROR
      WRITE (3,106) TIMES
      WRITE (3,108) (NSCALE(J), J = 1, N)
      WRITE (3,107) M
        DO 30 J = 1, N
        WRITE (3,110) J, X(J), COEFF(J)
   30   A(J) = X(J) * COEFF(J)
      HOLD = FUNCT (POINT)
      WRITE (3,100) HOLD
        IF (NUMPEN .GT. 0) WRITE (3,125) NUMPEN, PENAL
      DO 60 K = 1, MAXFN
        IOUT = KOUNT2 = 0
        DO 50 I = 1, N
          DO 40 J = 1, N
   40     A(J) = X(J) * COEFF(J)
C4      WRITE (3,110) I, X(I), COEFF(I)
      IF (X(I) .EQ. 0.0D0 .OR. NSCALE(I) .EQ. 0.0) THEN DO
        G = X(I)
        KOUNT2 = KOUNT2 + 1
        GO TO 50
      ENDIF
         IF (X(I) .GT. 0.0D0) THEN DO
          GLOW = X(I) * (1.0 - NSCALE(I))
          GHIGH = X(I) * (1.0 + NSCALE(I))
          IF (GLOW .LT. 1.0D-3 .AND. GLOW .GT. 0.0D0)  GLOW = -GLOW
         ELSE DO
          GLOW = X(I) * (1.0 + NSCALE(I))
          GHIGH = X(I) * (1.0 - NSCALE(I))
```

```
           IF (GHIGH .GT. -1.0D-3 .AND. GHIGH .LT. 0.0D0)  GHIGH = -GHIGH
          ENDIF
C4       WRITE (3,115) GLOW, GHIGH
         G = FMIN (GLOW, GHIGH, FUNCT, TOL)
         TOTAL = FUNCT (G)
C4       WRITE (3,120) G, SSQ
C4        IF (NUMPEN .GT. 0) WRITE (3,125) NUMPEN, PENAL
C4       IF (TOTAL .GT. HOLD) THEN DO
C4         PRINT,'BAD STEP?'
C4       ENDIF
C4       WRITE (3,160) TOTAL
   45     IF (DABS(X(I)-G) .GT. TOL) IOUT = 1
          IF (DABS((TOTAL-HOLD)/HOLD) .LT. TOL) KOUNT2 = KOUNT2 + 1
         HOLD = TOTAL
   50    X(I) = G
C
C  NORMALIZE X'S TO BETWEEN 1.0 & 10.0.  CORRECT CHANGE IN VALUE OF
C  COEFF SO PRODUCT IS SAME.
C
         DO 55 I = 1, N
         IF (X(I) .EQ. 0.0D0) GO TO 55
          WHILE (DABS(X(I)) .LT. 1.0D0) DO
           X(I) = X(I) * 10.0D0
           COEFF(I) = COEFF(I) / 10.0D0
          END WHILE
   55    CONTINUE
C
C  CHECK AND UPDATE CONVERGENCE CRITERIA - TERMINATE IF MET
C
       IF (IOUT .EQ. 0) KOUNT = KOUNT + 1
       IF (KOUNT. EQ. 2) THEN DO
        WRITE (3,135) TOL
        WRITE (3,145) K
        GO TO 70
       ENDIF
       IF(KOUNT2 .EQ. N) THEN DO
        WRITE (3,140) TOL
        WRITE (3,145) K
        GO TO 70
       ENDIF
C
   60 CONTINUE
      WRITE (3,147) MAXFN
   70   DO 80 J = 1, N
   80   WRITE (3,110) J, X(J), COEFF(J)
      REWIND 5
      WRITE (5,130) (X(I), I = 1, N)
      WRITE (5,130) (COEFF(I), I = 1, N)
      WRITE (3,150) TOTAL
      IF (NUMPEN .GT. 0) WRITE (3,155) NUMPEN
```

```
:6     B3 = PENLTY (1)
:7     WRITE (3,165)
:7     DO 85 I = 1, M
:7     WRITE (3,170) I, STRESS(I), RATE(I), Y(I), YAPROX(I)
:7 85  CONTINUE
       STOP
   95  FORMAT (20A4)
   98  FORMAT (1H1,'TITLE : ',20A4/)
  100  FORMAT (1H ,' AT START, FUNCT =', 1PD21.13//)
  105  FORMAT (1H-,'PROGRAM PARAMETERS:',/,T24,' TOL =',1PD9.2,/,T4,
     * 'ABS ERROR RAISED TO POWER:',1PD9.2,//,' PENALTY CONTROLS:'/,T15,
     * 'MULTIPLIER =',1PD12.5,/,T20,'POWER =',1PD9.2,/,T14,
     * 'ERROR LEVEL =',1PD12.5//)
  106  FORMAT (1H ,'TIMES =',1PD10.3)
  107  FORMAT (1H ,I3,' POINTS WERE OBSERVED'//)
  108  FORMAT (' SCALE VECTOR IS: ',6F6.2/)
  110  FORMAT (' ----- X(',I2,') =',1PD21.13,' * ',1PD8.1)
  115  FORMAT (' SEARCH RANGE  ',1PD12.5,' TO ',1PD12.5)
  120  FORMAT (1H ,'MINIMUM POINT @ G = ',1PD21.13,/,T15,
     *    'SSQ =',1PD21.13)
  125  FORMAT (I4,' CONSTRAINTS VIOLATED -- PENALTY =',1PD13.5)
  130  FORMAT (5D21.13)
  135  FORMAT (' CONVERGENCE OF ALL X''S TO WITHIN',1PD12.3)
  140  FORMAT (' DELTA FUNCT HAS BEEN LESS THAN',1PD12.5,' FOR 5 STEPS')
  145  FORMAT (1H-,'FINAL RESULT REACHED AFTER ',I3,' SERIES')
  147  FORMAT (1H-,'NO CONVERGENCE AFTER ',I3,' SERIES')
  150  FORMAT (1H-,'AT FINISH, FUNCT =',1PD21.13)
  155  FORMAT ('0AT FINISH, ',I3,' CONSTRAINTS VIOLATED:'/,
     * 1H ,T6,'#',T10,'STRESS(KSI)',T28,'RATE',
     * T38,'PRED. STRAIN',T52,'OBS. STRAIN',T69,'DIFF'/)
  160  FORMAT ('-FUNCT =',1PD21.13/)
  165  FORMAT(1H0,/,T4,'#',T15,'STRESS(KSI)',T34,'RATE',T45,'OBS. STRAIN'
     *   ,T60,'PRED. STRAIN'/)
  170  FORMAT (1H ,T3,I2,T10,4(1PD15.3))
       END
       DOUBLE PRECISION FUNCTION FUNCT(POINT)
       IMPLICIT REAL * 8 (A-H, O-Z)
       DIMENSION X(5),STRESS(60),Y(60),RATE(60),YAPROX(60),DIFF(60)
       COMMON STRESS, Y, RATE, YAPROX, M, SSQ, DIFF, TIMES
       COMMON /FUN1/ COEFF(5), TOTAL, UP, X, II
       SSQ = 0.0D0
       X(II) = POINT * COEFF(II)
C
C   CALCULATE PREDICTED STRAINS AND SUM OF ERRORS.
C
       DO 10 I = 1, M
         IF (STRESS(I) .EQ. 0.0D0) THEN DO
          YAPROX(I) = DIFF(I) = 0.0DC
          GO TO 10
         ENDIF
```

```
      TEMP2 = STRESS(I) / X(5) / RATE(I) ** X(4)
      TEMP3 = RATE(I) ** X(2)  *  STRESS(I) ** X(3)
      YAPROX(I) = TEMP2 + X(1) * TEMP3
C
C  APPLY WEIGHTING FACTORS TO RESIDUAL
C
      DIF = DABS(YAPROX(I) - Y(I))
C
      SSQ = SSQ + (DIF * TIMES) ** UP
      DIFF(I) = DIF
   10 CONTINUE
      TOTAL = SSQ + PENLTY (0)
      FUNCT = TOTAL
C3    WRITE (3,100) POINT, FUNCT
      RETURN
  100 FORMAT (1H ,'AT FUNCT, POINT =',1PD21.13,' FUNCT =',1PD21.13)
      END
      DOUBLE PRECISION FUNCTION PENLTY (IDUM)
      IMPLICIT REAL * 8 (A-H, O-Z)
      DIMENSION STRESS(60),Y(60),RATE(60),YAPROX(60),DIFF(60)
      COMMON STRESS, Y, RATE, YAPROX, M, SSQ, DIFF, TIMES
      COMMON /WEIGHT/ R, Z, ERROR, KOUNT, HOLD
      HOLD = 0.0D0
      KOUNT = 0
C
C  CALCULATE PENALTY IF ABS. ERROR GREATER THAN SPECIFIED MAX.
C
      DO 10 I = 1, M
      B3 = DIFF(I)
      IF (B3 .GT. ERROR) THEN DO
       HOLD = HOLD + (B3 * TIMES) ** Z
       KOUNT = KOUNT + 1
        IF (IDUM .EQ. 1) THEN DO
         WRITE (3,100) KOUNT,STRESS(I),RATE(I),YAPROX(I),Y(I),B3
        ENDIF
      ENDIF
   10 CONTINUE
      PENLTY = HOLD = HOLD * R
  100 FORMAT (1H ,I5,5(1PD14.5))
      RETURN  ;   END
C$OPTIONS NOLIST
      DOUBLE PRECISION FUNCTION FMIN(AX,BX,F,TOL)
      DOUBLE PRECISION AX,BX,F,TOL
      DOUBLE PRECISION  A,B,C,D,E,EPS,XM,P,Q,R,TOL1,TOL2,U,V,W
      DOUBLE PRECISION  FU,FV,FW,FX,X
      DOUBLE PRECISION  DABS,DSQRT,DSIGN
      COMMON /SEARCH/ EPS
      C = 0.5D0*(3. - DSQRT(5.0D0))
      A = AX
      B = BX
```

```
      V = A + C*(B - A)
      W = V
      X = V
      E = 0.0D0
      FX = F(X)
      FV = FX
      FW = FX
   20 XM = 0.5D0*(A + B)
      TOL1 = EPS*DABS(X) + TOL/3.0D0
      TOL2 = 2.0D0*TOL1
      IF (DABS(X - XM) .LE. (TOL2 - 0.5D0*(B - A))) GO TO 90
      IF (DABS(E) .LE. TOL1) GO TO 40
      R = (X - W)*(FX - FV)
      Q = (X - V)*(FX - FW)
      P = (X - V)*Q - (X - W)*R
      Q = 2.0D00*(Q - R)
      IF (Q .GT. 0.0D0) P = -P
      Q =  DABS(Q)
      R = E
      E = D
   30 IF (DABS(P) .GE. DABS(0.5D0*Q*R)) GO TO 40
      IF (P .LE. Q*(A - X)) GO TO 40
      IF (P .GE. Q*(B - X)) GO TO 40
      D = P/Q
      U = X + D
      IF ((U - A) .LT. TOL2) D = DSIGN(TOL1, XM - X)
      IF ((B - U) .LT. TOL2) D = DSIGN(TOL1, XM - X)
      GO TO 50
   40 IF (X .GE. XM) E = A - X
      IF (X .LT. XM) E = B - X
      D = C*E
   50 IF (DABS(D) .GE. TOL1) U = X + D
      IF (DABS(D) .LT. TOL1) U = X + DSIGN(TOL1, D)
      FU = F(U)
      IF (FU .GT. FX) GO TO 60
      IF (U .GE. X) A = X
      IF (U .LT. X) B = X
      V = W
      FV = FW
      W = X
      FW = FX
      X = U
      FX = FU
      GO TO 20
   60 IF (U .LT. X) A = U
      IF (U .GE. X) B = U
      IF (FU .LE. FW) GO TO 70
      IF (W .EQ. X) GO TO 70
      IF (FU .LE. FV) GO TO 80
      IF (V .EQ. X) GO TO 80
```

```
      IF (V .EQ. W) GO TO 80
      GO TO 20
   70 V = W
      FV = FW
      W = U
      FW = FU
      GO TO 20
   80 V = U
      FV = FU
      GO TO 20
   90 FMIN = X
      RETURN
      END
DATA
 5
  3.85D0          -2.35D0         4.52D0     2.106D0     1.694D0
 1.0D-68          1.0D0          1.0D1      1.0D-2      1.0D3
```

```
C***********************************************************************
C                                                                      *
C PROGRAM NAME : SOLVALL                                               *
C WRITTEN BY : J.F. BRANDEAU                                           *
C COMPILER(S) : WATFIV (DOUBLE PRECISION)                              *
C         ---------------------------------------------------          *
C PURPOSE : TO CONVERT EQUATION STRAIN = F(RATE, STRESS) TO EQUATION   *
C STRESS = F(RATE, STRAIN) USING SUBROUTINE ZEROIN TO SOLVE FOR ROOT OF*
C THE EQUATION.  CAN HANDLE UP TO SIX SETS OF COEFFICIENTS X TO PRODUCE*
C UP TO SIX DATA PAIRS FOR SAS TO GRAPH ON A SINGLE GRAPH.             *
C         ---------------------------------------------------          *
C VARIABLES :                                                          *
C                                                                      *
C KGRAPS : NUMBER OF SETS OF COEFFICIENTS TO BE TAKEN FROM DATA.       *
C                                                                      *
C RATE : STRAIN RATE TO BE USED.                                       *
C                                                                      *
C KEPS : NUMBER OF POINTS TO BE GENERATED FOR EACH UNIQUE VECTOR X.    *
C                                                                      *
C TOL : CONVERGENCE CRITERION FOR ZEROIN.                              *
C                                                                      *
C A, B, N, D, C (I) - SET OF COEFFICIENTS FOR EACH CURVE.  THE I'TH    *
C  ELEMENT OF EACH CONSITITUTES ONE SET OF COEFFICIENTS X.             *
C                                                                      *
C SIGMAX : MAXIMUM VALUE OF STRESS (KSI) EXPECTED FOR EACH SET OF      *
C  COEFFICIENTS X.  THIS IS THE HIGH LIMIT SENT TO ZEROIN.  THESE MAY  *
C  BE ADJUSTED DOWNWARD BY THE PROGRAM IF NEEDED TO PREVENT UNDER/OVER *
C  FLOWS.  IF THE VALUE OF SIGMAX(I) IS NOT HIGH ENOUGH FOR COEFFICIENT*
C  X(I), THE CURVE WILL BE FLATTENED AT THE HIGHER STRESSES.           *
C                                                                      *
C EPSMAX : MAXIMUM VALUE OF STRAIN FOR WHICH EACH CURVE IS TO BE       *
C  EVALUATED.                                                          *
C         ---------------------------------------------------          *
C I/O REQUIREMENTS :                                                   *
C                                                                      *
C FILE #1 : ALL INPUT FROM ABOVE, FOLLOWING $DATA CARD.                *
C FILE #6 : OUTPUT OF STRESS-STRAIN PAIRS FOR USE BY SAS (LRECL=130).  *
C         ---------------------------------------------------          *
C                                                                      *
C CCOMP OPTIONS (FORM C$OPTIONS CCOMP=?????) :                         *
C                                                                      *
C 4 : OUTPUT OF RETURNED STRESS VALUES.                                *
C                                                                      *
C***********************************************************************
C$OPTIONS CCOMP=0
      IMPLICIT REAL * 8 (A-H, N, O-Z)
      EXTERNAL FUNCT
      DIMENSION A(6), B(6), C(6), D(6), N(6), SIGMAX(6), EPSMAX(6)
      DIMENSION EPS(50,6), STRESS(50,6)
      COMMON A, B, C, D, N, TLOG, RATE, KOUNT, J, EPS, RATEB, RATED
```

```
      COMMON /SUB2/ GEPS
C
C CALCULATE MACHINE EPSILON
C
      GEPS = 1.0D0
    4 GEPS = GEPS/2.0D0
      TOL1 = 1.0D0 + GEPS
      IF (TOL1 .GT. 1.0D0) GO TO 4
C
      READ (1,*) KGRAFS, RATE, KEPS, TOL
      TEPS = 0.0D0
      DO 10 I = 1, KGRAFS
      READ (1,*) A(I), B(I), N(I), D(I), C(I), SIGMAX(I), EPSMAX(I)
C
C CHECK FOR POSSIBLE OVERFLOW FOR HIGH N'S
C ALTER SIGMAX DOWNWARD IF NECESSARY
C
      RATEB = DLOG10(RATE ** B(I))
    6 AX = N(I) * DLOG10 (SIGMAX(I)) + RATEB
       IF (AX. GT. 75.0D0) THEN DO
       SIGMAX(I) = SIGMAX(I) - 0.5D0
       GO TO 6
      ENDIF
   10 CONTINUE
C
C  BEGIN MAIN LOOP
C
      DO 30 KOUNT = 1, KGRAFS
      STRESS(1,KOUNT) = EPS(1,KOUNT) = 0.0D0
      BX = SIGMAX(KOUNT)
      DEPS = EPSMAX(KOUNT) / DFLOAT(KEPS-1)
      RATED = RATE ** D(KOUNT)
      RATEB = RATE ** B(KOUNT)
C
C  WATCHING FOR VALUES OF N THAT WILL CAUSE OVERFLOW OR UNDERFLOW
C
      IF (A(KOUNT) .EQ. 0.0D0) THEN DO
       TLOG = -80.0D0
      ELSE DO
       TLOG = DLOG10 (A(KOUNT))
      ENDIF
C
C  INNER LOOP FOR EACH SOLUTION
C
        DO 20 J = 2, KEPS
        EPS(J,KOUNT) = DFLOAT(J-1) * DEPS
        AX = STRESS(J-1,KOUNT)
        IF (J .EQ. 2) AX = ( C(KOUNT) * EPS(J,KOUNT) * RATED) / 1.3D0
        STRESS(J,KOUNT) = ZEROIN (AX, BX, FUNCT, TOL)
C4      PRINT,'FROM ZEROIN, STRESS',J,' =',STRESS(J,KOUNT)
```

```
   20   CONTINUE
   30 CONTINUE
      DO 40 I = 1, KEPS
   40 WRITE (6,100) (EPS(I,J), STRESS(I,J), J = 1, KGRAFS)
   50 STOP
  100 FORMAT (10(1PD13.5))
      END
      DOUBLE PRECISION FUNCTION FUNCT(STRESS)
      IMPLICIT REAL * 8 (A-H, N, O-Z)
      DIMENSION A(6), B(6), C(6), D(6), N(6), EPS(50,6)
      COMMON A, B, C, D, N, TLOG, RATE, KOUNT, J, EPS, RATEB, RATED
      IF (STRESS. EQ. 0.0D0) THEN DO
       FUNCT = -EPS(J,KOUNT)
       RETURN
      ENDIF
      TEMP1 = STRESS / (C(KOUNT) * RATED)
      IF (STRESS .GT. 1.0D-1) GO TO 10
      IF ((N(KOUNT) * DLOG10(STRESS)) . LT. -60.0D0) THEN DO
       HOLD = -25.0D0
       GO TO 15
      ENDIF
   10 TEMP2 = STRESS ** N(KOUNT)  *  RATEB
      HOLD = DLOG10(TEMP2)
   15 IF ((HOLD + TLOG) .LT. -17.0D0) THEN DO
       FUNCT = TEMP1 - EPS(J,KOUNT)
      ELSE DO
       FUNCT = (TEMP1 + A(KOUNT) * TEMP2) - EPS(J,KOUNT)
      ENDIF
25    RETURN    :    END
$OPTIONS NOLIST
      DOUBLE PRECISION FUNCTION ZEROIN(AX,BX,F,TOL)
      DOUBLE PRECISION AX,BX,F,TCL
      DOUBLE PRECISION  A,B,C,D,E,EPS,FA,FB,FC,TOL1,XM,P,Q,R,S
      DOUBLE PRECISION  DABS,DSIGN
      COMMON /SUB2/ EPS
      A = AX
      B = BX
      FA = F(A)
      FB = F(B)
   20 C = A
      FC = FA
      D = B - A
      E = D
   30 IF (DABS(FC) .GE. DABS(FB)) GO TO 40
      A = B
      B = C
      C = A
      FA = FB
      FB = FC
      FC = FA
```

```
   40 TOL1 = 2.0D0*EPS*DABS(B) + 0.5D0*TOL
      XM = .5*(C - B)
      IF (DABS(XM) .LE. TOL1) GO TO 90
      IF (FB .EQ. 0.0D0) GO TO 90
      IF (DABS(E) .LT. TOL1) GO TO 70
      IF (DABS(FA) .LE. DABS(FB)) GO TO 70
      IF (A .NE. C) GO TO 50
      S = FB/FA
      P = 2.0D0*XM*S
      Q = 1.0D0 - S
      GO TO 60
   50 Q = FA/FC
      R = FB/FC
      S = FB/FA
      P = S*(2.0D0*XM*Q*(Q - R) - (B - A)*(R - 1.0D0))
      Q = (Q - 1.0D0)*(R - 1.0D0)*(S - 1.0D0)
   60 IF (P .GT. 0.0D0) Q = -Q
      P = DABS(P)
      IF ((2.0D0*P) .GE. (3.0D0*XM*Q - DABS(TOL1*Q))) GO TO 70
      IF (P .GE. DABS(0.5D0*E*Q)) GO TO 70
      E = D
      D = P/Q
      GO TO 80
   70 D = XM
      E = D
   80 A = B
      FA = FB
      IF (DABS(D) .GT. TOL1) B = B + D
      IF (DABS(D) .LE. TOL1) B = B + DSIGN(TOL1, XM)
      FB = F(B)
      IF ((FB*(FC/DABS(FC))) .GT. 0.0D0) GO TO 20
      GO TO 30
   90 ZEROIN = B
      RETURN
      END
$DATA
5   1.0D0   25        1.0D-7
5.1183D-13    -3.74D-1  6.5342D0  6.707D-2    3.5514D3    50.0D0   .016D0
36.777D-13 -4.1267D-1 7.6617D0  5.67034D-2 2.20498D3 25.0D0 .0060D0
3.052D-13 -1.426D-1 7.632D0  6.287D-2    2.284D3    20.0    .0070D0
3.709D-68  -2.3357D0  45.2472D0  1.7977D-2  1.694D3 45.0D0 .036D0
).0D0  1.0D0  1.0D0   1.34946D-2  2.6354D2  15.0D0    .030D0
```

```
:*********************************************************************
:                                                                    *
: PROGRAM NAME : BAKSOLV                                             *
: WRITTEN BY : J.F. BRANDEAU                                         *
: COMPILER(S) : WATFIV (DOUBLE PRECISION)                            *
:         ----------------------------------------------------       *
: PURPOSE : CONVERT EQUATION STRAIN = F(RATE, STRESS) TO EQUATION    *
: STRESS = F(RATE, STRAIN) BY USING SUBROUTINE ZEROIN TO SOLVE FOR ROOT*
: OF EQUATION.  PROGRAM CHECKS RATE TO DETERMINE EACH CHANGE AND     *
: PREVENT UNDER / OVER FLOW.  6 UNIQUE RATES ARE ALLOWED IN THE INPUT *
: LIST FROM FILE #4.                                                 *
:         ----------------------------------------------------       *
: VARIABLES :                                                        *
:                                                                    *
: SIGMAX : GREATER THAN MAXIMUM VALUE OF STRESS EXPECTED FOR EACH     *
:  OBSERVED RATE.  THIS IS THE UPPER LIMIT FOR ROOT SEARCH, AND IS   *
:  ADJUSTED TO PREVENT OVER /UNDER FLOW.  IF THIS IS TOO LOW THE CURVE *
:  FOR THAT STRAIN RATE WILL BE FLATTENED AT THE TOP.                *
:                                                                    *
: TOL : CONVERGENCE CRITERION FOR ZEROIN.                            *
:                                                                    *
: X & COEFF : MANTISSA AND EXPONENT OF COEFFICIENT VECTOR.  PROGRAM  *
:  COMBINES BOTH INTO X.                                             *
:         ------------------------------------------------------     *
: I / O REQUIREMENTS :                                               *
:                                                                    *
: FILE #4 : OBSERVED VALUES OF DATA AS USED FOR OTHER PROGRAMS.      *
:           TITLE MUST BE ON FIRST RECORD, FOLLOWED BY ONE OBSERVATION *
:           PER RECORD; STRAIN RATE, STRAIN AND STRESS (IN ORDER).   *
: FILE #5 : X & COEFF.  X IS THE MANTISSA AND COEFF THE EXPONENT OF  *
:           THE VARIABLE COEFFICIENTS.  PRODUCT X * COEFF SHOULD EQUAL *
:           THE COEFFICIENT.                                         *
: FILE #6 : OUTPUT OF POINTS FOR USE BY SAS.                         *
:                                                                    *
:*********************************************************************
      IMPLICIT REAL * 8 (A-H, O-Z)
      EXTERNAL FUNCT
      DIMENSION X(5), SIGMAX(6), COEFF(5)
      COMMON X, RATE, EPS, RATEB, RATED
      INTEGER TITLE(20)
      DATA SIGMAX / 50.0, 50.0, 50.0, 50.0, 50.0, 50.0/
      TOL = 1.0D-7
      READ (5,*) X
      READ (5,*) COEFF
      READ (4,200) TITLE
      WRITE (3,300) TITLE
      DO 5 J = 1, 5
      X(J) = X(J) * COEFF(J)
    5 WRITE (3,100) X(J)
      RATE1 = 0.0
```

```
      KP = 0
   10 READ (4,*,END=50) RATE, EPS, STRESS
:
:  HAS STRAIN RATE CHANGED IN INPUT LIST?
:
      IF (RATE .NE. RATE1) THEN DO
       RATE1 = RATE
       KP = KP + 1
       KOUNT = 0
:
:  CORRECT SIGMAX TO PREVENT OVER / UNDER FLOW.
:
      RATEB = DLOG10 (RATE ** X(2))
   15 AX = X(3) * DLOG10 (SIGMAX(KP)) + RATEB
       IF (AX .GT. 75.0D0) THEN DO
        SIGMAX(KP) = SIGMAX(KP) - 0.5D0
        GO TO 15
       ENDIF
:
      RATEB = RATE ** X(2)
      RATED = RATE ** X(4)
      ENDIF
      KOUNT = KOUNT + 1
      IF (EPS .EQ. 0.0D0) THEN DO
       SIG1 = 0.0D0
      ELSE DO
       AX = SIG1
       BX = SIGMAX(KP)
       IF (KOUNT .EQ. 2) AX = STRESS / 1.3
       SIG1 = ZEROIN ( AX, BX, FUNCT, TOL)
      ENDIF
      WRITE (6,400) RATE, STRESS, EPS, SIG1
      GO TO 10
   50 STOP
  100 FORMAT (1H ,1PD21.13)
  200 FORMAT (20A4)
  300 FORMAT (1H ,'TITLE : ',20A4)
  400 FORMAT (1H ,5(1PD13.5))
      END
      DOUBLE PRECISION FUNCTION FUNCT(STRESS)
      IMPLICIT REAL * 8 (A-H, O-Z)
      DIMENSION X(5)
      COMMON X, RATE, EPS, RATEB, RATED
      TEMP1 = STRESS / X(5) / RATED
      FUNCT = TEMP1 + X(1) * RATEB * STRESS ** X(3) - EPS
      RETURN
      END
:$OPTIONS NOLIST
      DOUBLE PRECISION FUNCTION ZEROIN(AX,BX,F,TOL)
      DOUBLE PRECISION AX,BX,F,TOL
```

```
      DOUBLE PRECISION  A,B,C,D,E,EPS,FA,FB,FC,TOL1,XM,P,Q,R,S
      DOUBLE PRECISION  DABS,DSIGN
      EPS = 1.0D0
   10 EPS = EPS/2.0D0
      TOL1 = 1.0D0 + EPS
      IF (TOL1 .GT. 1.0D0) GO TO 10
      A = AX
      B = BX
      FA = F(A)
      FB = F(B)
   20 C = A
      FC = FA
      D = B - A
      E = D
   30 IF (DABS(FC) .GE. DABS(FB)) GO TO 40
      A = B
      B = C
      C = A
      FA = FB
      FB = FC
      FC = FA
   40 TOL1 = 2.0D0*EPS*DABS(B) + 0.5D0*TOL
      XM = .5*(C - B)
      IF (DABS(XM) .LE. TOL1) GO TO 90
      IF (FB .EQ. 0.0D0) GO TO 90
      IF (DABS(E) .LT. TOL1) GO TO 70
      IF (DABS(FA) .LE. DABS(FB)) GO TO 70
      IF (A .NE. C) GO TO 50
      S = FB/FA
      P = 2.0D0*XM*S
      Q = 1.0D0 - S
      GO TO 60
   50 Q = FA/FC
      R = FB/FC
      S = FB/FA
      P = S*(2.0D0*XM*Q*(Q - R) - (B - A)*(R - 1.0D0))
      Q = (Q - 1.0D0)*(R - 1.0D0)*(S - 1.0D0)
   60 IF (P .GT. 0.0D0) Q = -Q
      P = DABS(P)
      IF ((2.0D0*P) .GE. (3.0D0*XM*Q - DABS(TOL1*Q))) GO TO 70
      IF (P .GE. DABS(0.5D0*E*Q)) GO TO 70
      E = D
      D = P/Q
      GO TO 80
   70 D = XM
      E = D
   80 A = B
      FA = FB
      IF (DABS(D) .GT. TOL1) B = B + D
      IF (DABS(D) .LE. TOL1) B = B + DSIGN(TOL1, XM)
```

```
      FB = F(B)
      IF ((FB*(FC/DABS(FC))) .GT. 0.0D0) GO TO 20
      GO TO 30
   90 ZEROIN = B
      RETURN
      END
```

```
C***********************************************************************
C                                                                      *
C PROGRAM NAME : MAP2                                                  *
C WRITTEN BY : J.F. BRANDEAU                                           *
C COMPILER(S) : WATFIV  (DOUBLE PRECISION)                             *
C          -------------------------------------------------           *
C PURPOSE : USE A SET OF OBSERVED POINTS AND COEFFICIENT VECTOR X TO   *
C MATCH A PREDICTED VALUE OF STRAIN = F(STRESS, STRAIN RATE) TO EACH   *
C OBSERVED POINT.  READS FROM EXTERNAL FILE FOR COEFFICIENTS AND OBS-  *
C ERVED VALUES, AND CAN READ COEFFICIENTS FROM TRAILING LIST (THESE    *
C OVERRIDE EXTERNAL VALUES).  SENDS OBSERVED POINTS AND PREDICTED      *
C POINTS TO A FILE TO BE USED BY SAS.  CAN ALSO DO SENSITIVITY ANALYSIS*
C WITHOUT CHANGING EXTERNAL VALUES OF COEFFICIENTS.                    *
C          -------------------------------------------------           *
C VARIABLES :                                                          *
C                                                                      *
C X & COEFF : MANTISSAS AND EXPONENTS OF COEFFICIENT VECTOR.  THESE ARE*
C  COMBINED INTO X IN THE PROGRAM.                                     *
C                                                                      *
C DELTA : FRACTIONAL CHANGE IN VARIABLE K FOR SENSITIVITY ANALYSIS.    *
C  SET THIS EQUAL TO ZERO TO GET TRUE COEFFICIENTS.                    *
C                                                                      *
C K : VARIABLE THAT WILL BE ALTERED BY AMOUNT (DELTA * X(K)).  MUST BE *
C  BETWEEN 1 AND 5 ALWAYS.                                             *
C                                                                      *
C STRESS : OBSERVED VALUES OF STRESS (KSI).                            *
C                                                                      *
C RATE : OBSERVED VALUES OF STRAIN RATE (1/SEC).                       *
C                                                                      *
C EPS : OBSERVED VALUES OF STRAIN (IN/IN).                             *
C                                                                      *
C EPS1 : PREDICTED VALUE OF STRAIN RATE (IN/IN).                       *
C          -------------------------------------------------           *
C I/O REQUIREMENTS :                                                   *
C                                                                      *
C FILE #1 : (OPTIONAL) X & COEFF VECTORS ON TWO RECORDS.               *
C FILE #4 : OBSERVED VALUES OF RATE, EPS, STRESS IN THIS ORDER.  THE   *
C           FIRST CARD MUST BE A TITLE.  THE REMAINING CARDS CONTAIN   *
C           ONE OBSERVATION EACH.                                      *
C FILE #5 : X & COEFF VECTORS ON TWO RECORDS.  WILL ALWAYS BE READ.    *
C          -------------------------------------------------           *
C OPTIONS (FORM C$OPTIONS CCOMP=??????) :                              *
C                                                                      *
C 3 : READ FROM DATA CARDS AT END OF PROGRAM LIST, FOLLOWING $DATA CARD*
C                                                                      *
C***********************************************************************
C$OPTIONS CCOMP=0
      IMPLICIT REAL * 8 (A-H, N, O-Z)
      DIMENSION X(5), COEFF(5)
      INTEGER TITLE(20)
```

```
      READ (5,*) X
      READ (5,*) COEFF
C3    READ (1,*) X
C3    READ (1,*) COEFF
      DELTA = 0.0D0
      K = 1
      X(K) = X(K) * (1.0D0 + DELTA)
      DO 5 J = 1, 5
      X(J) = X(J) * COEFF(J)
    5 WRITE (3,75) X(J)
      READ (4,200) TITLE
      WRITE (3,300) TITLE
   10 READ (4,*,END=50) RATE, EPS, STRESS
C
C  CALCULATE PREDIICTED STRAIN
C
      TEMP1 = STRESS / X(5) / RATE ** X(4)
      TEMP2 = STRESS ** X(3)  *  RATE ** X(2)
      EPS1 = TEMP1 + X(1) * TEMP2
C
      WRITE (6,100) RATE, STRESS, EPS, EPS1
      GO TO 10
   50 STOP
   75 FORMAT (1H ,1PD20.12)
  100 FORMAT (4D25.13)
  200 FORMAT (20A4)
  300 FORMAT (1H ,20A4)
      END
5DATA
5.63      -0.589    6.19      3.20     3.10
1.0D-13    1.0D0     1.0D0     1.0D-2   1.0D3
```

```
'/FORCE1 JOB DU.D08.AQ0221,BRANDEAU,T=(,10),M=(2,0)
'/  EXEC WATFIV
'/GO.FT09F001 DD DSN=DU.D08.AQ0221.BRANDEAU.DATA.ONE,DISP=SHR
'/GO.FT08F001 DD DSN=DU.D08.AQ0221.BRANDEAU.ATBDATA.ONE,DISP=SHR
'/GO.SYSIN DD *
'JOB
:$OPTIONS CCOMP=2,NOEXT,NOCHECK,DEC
: *******************************************************************
:  SUBROUTINE FORCE IN MAIN PROGRAM FORM
:  THIS PROGRAM ANALYZES THE ATB OUTPUT DATA FOR DATA NEEDED IN THE
:  OPERATION OF THE BREAK PROGRAM.
:    1)   THE TOP OF EACH TIME INCREMENT DATA SET IS FOUND
:    2)   ROTATIONS AND ANGULAR VELOCITIES FOR TIME STEP ARE FOUND
:    3)   DISPLACEMENTS AND LINEAR VELOCITIES ARE FOUND
:    4)   JOINT FORCES AND TORQUES ARE FOUND
:    5)   DATA IS WRITTEN TO PRINTER AND/OR DISK
:          A. OUTPUT TO DISK IS IN UNFORMATTED FORM
:          B. FOR EACH TIME INCREMENT, THE TIME (MS) AND DATA FOR EACH
:             LIMB IS OUTPUT.  EACH LIMB
:             DATA SET FOR THAT TIME INCREMENT IS ON A RECORD, PRECEDED
:             BY THE IDENTIFYING NUMBER FOR THAT LIMB (1 - 8).
: *******************************************************************
:  KCON IS THE NUMBER OF "OTHER CONSTRAINT FORCES"
:    -1 = NONE
:    >0 = NUMBER OF ROWS OF DATA TO BE FOUND FOR EACH TIME STEP
:
: POSIT = POINT OF ATTACHMENT RELATIVE TO C.G. OF SEGMENT IN SEGMENT
:    LOCAL Z-AXIAL COORDINATES.
: CONSTRN(I,J) = FORCES IN INERTIAL COORDINATES (J = 1, 3) FOR SEGMENT I
: ( I = 1, KCON)
:
      CHARACTER *4 IDUM,IFLAG
      REAL D(8,30), VEH(18), POSIT(24), CNSTRN(8,3)
      REAL * 8 DD(3), DA(3)
      INTEGER KT3(8)
      DATA KT3 / 8 * 1 /, CNSTRN /24 * 0.0/
      NT = 31    ;   DLT = 0.01
      IW = 3    ;  IR = 1  ;   IDISK1 = 9    ;   IDISK2 = 8
      WRITE (IW,100)
      JP = 1
      READ (1,*) KCON
      KCON4 = KCON * 4
      IF (KCON4 .EQ. 0) KCON4 = 1
      POSIT(1) = 0
      DO 2 I = 1, KCON4, 4
        READ (1,*) POSIT(I), POSIT(I+1), POSIT(I+2), POSIT(I+3)
    2 KT3(POSIT(I)) = 3
:2    WRITE (IDISK2) NT, DLT, KCON4, (POSIT(I), I = 1, KCON4)
      DO 4 J = 1, 200
        READ (IDISK1,400) IDUM
```

```
      IF (IDUM .EQ. ' 5 H') GO TO 6
    4 CONTINUE
    6 DO 8 J = 1, 10
        READ (IDISK1,420) I, W, X, Y, Z, A, B, C
        IF (J .EQ. 3 .OR. J .EQ .6) GO TO 8
C3      WRITE (IW, 120) I, W, X, Y, Z, A, B, C
C
C  OUTPUT TO DISK IN X-AXIAL COORDINATES AND CONSECUTIVE SEGMENT NUMBERS
C
C2      WRITE (IDISK2) JP, W, Z, X, Y, C, A, B
        JP = JP + 1
    8 CONTINUE
      WRITE (IW, 130) (POSIT(KK) , KK = 1, KCON4)
      IP = 0
              DO 80 I = 1, 1000
              ON ERROR GOTO 75
        IF (IP .GE. NT) THEN DO
          PRINT,IP,' TIME STEPS FOUND'
          STOP    ;    ENDIF
C
C  FIND TOP OF DATA SET FOR EACH TIME INCREMENT
C
   13 READ (IDISK1 ,900, END = 90) IFLAG
      IF (IFLAG .NE. 'MAIN') GO TO 13
      BACKSPACE IDISK1
      READ (IDISK1, 905) TIME
      TIME = TIME / 1000.
C
C  FIND ROTATIONS , ANGULAR VEL. AND ANGULAR ACC. FOR THIS TIME STEP
C
      IP = IP + 1
      DO 20 II = 1, 15
      READ (IDISK1, 1000) IDUM
      IF (IDUM .NE. 'H    ') GO TO 20
      JP = 1
        DO 15 J = 1, 10
        READ (IDISK1 ,1100) (D(JP,KK),KK = 7,9), (DD(KK),KK = 1,3),
     *     (DA(KK), KK = 1,3)
        IF (J .EQ. 3 .OR. J .EQ. 6) GO TO 15
          DO 14 K4 = 1, 3
          D(JP,K4+18) = SNGL (DA(K4))
   14     D(JP,K4+9) = SNGL (DD(K4))
        JP = JP + 1
   15   CONTINUE
      READ (IDISK1,800) (VEH(KK), KK = 7, 12)
      GO TO 22
   20 CONTINUE
C
C  FIND LINEAR POSITION, VELOCITY AND ACCELERATION FOR THIS TIME STEP
C
```

```
   22 DO 40 II = 1, 15
         READ (IDISK1 ,1000) IDUM
         IF (IDUM .NE. 'H   ') GO TO 40
         JP = 1
         DO 25 J = 1, 10
           READ (IDISK1,2000) (D(JP,KK),KK = 1, 6), (D(JP,KK), KK = 22, 24)
           IF ( J .EQ. 3 .OR. J . EQ. 6 ) GO TO 25
           JP = JP + 1
   25 CONTINUE
         READ (IDISK1,1000) IDUM
         READ (IDISK1,2000) (VEH(KK), KK = 1, 6)
         GO TO 50
   40 CONTINUE
C
C     FIND U1 & U2 ARRAYS
C
   50 DO 42 II = 1, 25
           READ (IDISK1,400) IDUM
   42 IF (IDUM .EQ. ' 5 H') GO TO 44
   44 JP = 1
         DO 46 JJ = 1, 10
           READ (IDISK1,2000) (D(JP,KK), KK = 25, 30)
           IF (JJ .EQ. 3 .OR. JJ .EQ. 6) GO TO 46
           JP = JP + 1
   46 CONTINUE
C
C     FIND FORCES AND TORQUES FOR THIS TIME INCREMENT
C
         DO 70 IJ = 1, 50
           READ (IDISK1 ,1000) IDUM
           IF (IDUM .NE. 'HP  ') GO TO 70
           JP = 1
           DO 60 JJ = 1, 10
             READ (IDISK1 ,4000) (D(JP,KK), KK = 13, 18)
             IF (JJ .EQ. 3 .OR. JJ .EQ. 6) GO TO 60
             JP = JP + 1
   60      CONTINUE
           READ(IDISK1,4000) (VEH(KK), KK = 13, 18)
           GO TO 71
   70 CONTINUE
   71 DO 74 II = 1, 20
           READ (IDISK1,400) IDUM
           IF (IDUM .NE. ' NO.') GO TO 74
           READ (IDISK1,400) IDUM
           DO 72 IJ = 1, KCON
   72      READ (IDISK1, 500) (CNSTRN(POSIT(4*IJ-3),IN), IN = 1, 3)
         GO TO 75
   74 CONTINUE
   75 CONTINUE
C
```

```
C  OUTPUT RESULTS
C  C2 = DISK OUTPUT (UNIT =  IDISK2)
C  C3 = LINE PRINTER (UNIT =  IW)
C2    WRITE (IDISK2) TIME, VEH
C3    PRINT,'TIME =',TIME,' VEHICLE =', VEH
C2    PRINT,'TIME =',TIME,' SEC'
        DO 85 J = 1, 8
        K3 = KT3(J)
C2      WRITE (IDISK2) J,K3,(D(J,KK), KK=1,30),(CNSTRN(J,KK),KK=1,K3)
C3      WRITE (IW,808) J,K3,(D(J,KK), KK=1,30),(CNSTRN(J,KK),KK=1,K3)
   85   CONTINUE
C
   80         CONTINUE
   90 PRINT,'END OF FILE REACHED ON UNIT',IDISK1
      PRINT,IP,' TIME STEPS FOUND'
      STOP
C
C  INPUT FORMATS
  130 FORMAT (' EXTRA FORCE LOCATIONS :',F5.0, 3F12.3)
  400 FORMAT (1X,A4)
  500 FORMAT (25X, 3F15.5)
  420 FORMAT (I3, 13X, F7.3, 3X, 3F11.5, 5X, 3F8.3)
  800 FORMAT (/, 11X, 3F9.4, 4X, 3(F7.3, 7X))
  900 FORMAT (7X, A4)
  905 FORMAT (57X, F8.3)
 1000 FORMAT (4X, A4)
 1100 FORMAT (11X, 3F9.4, 3X, 3D14.5, 3X, 3D14.5)
 2000 FORMAT (11X, 3F11.4, 3X, 3F12.5, 3X, 3F14.5)
 4000 FORMAT (15X, 3F11.4, 3X, 3F12.5)
C
C  OUTPUT FORMATS
  100 FORMAT (1H1, 10X, 'FORCES FROM ATB MODEL'/)
C3120 FORMAT (' I=', I2, ' W=', F7.3,  ' XYZ=', 3F10.5,  ' ABC=', 3F7.3)
C3808 FORMAT (1H , I2, I3, 2X, (T11,6(E16.7, 3X)))
      END
$DATA
2
2   0.0   0.0   0.0
4   0.0   0.0   0.0
C$STOP
C$END
/*
//*PW=BONE
//
```

```
//FORCE2 JOB DU.D08.AQ0221,BRANDEAU,T=(,10),P=45,M=(2,0)
//   EXEC   WATFIV
//GO.FT09F001 DD DSN=DU.D08.AQ0221.BRANDEAU.DATA.TWO,DISP=SHR
//GO.FT08F001 DD DSN=DU.D08.AQ0221.BRANDEAU.ATBDATA.TWO,DISP=SHR
//GO.SYSIN DD *
$JOB
:$OPTIONS NOEXT,CCOMP=2,NOCHECK,DEC
:$OPTIONS NOLIST
      CHARACTER * 10 IDUM, DUMB
      CHARACTER * 3 ISEG1, ISEG2
      REAL PANEL (32,8,3,6), SEGMET (32,8,3,6), X1 (3), X2 (3)
      INTEGER CHEKP (32,8), CHEKS (32,8), ID(3), UCOUNT, NP(4)
      CHARACTER ITEST*3(8)/'RUL','RLL','LUL','LLL','RUA','RLA','LUA'
     *   ,'LLA'/
      DATA CHEKP,CHEKS / 512  * 0/
      DATA PANEL,SEGMET /9216 * 0.0/
      DATA NP / 58, 92, 92, 92 /
:*********************************************************************
:  EXPLANATION OF VARIABLES --
:  PANEL (I,J,K,L) AND SEGMET (I,J,K,L) CONTAIN CONTACT DATA
:    I = TIME STEP
:    J = LIMB OR MEMBER NUMBER
:    K = CONTACT NUMBER
:    L = 1 TO 6
:       L = 1 IS THE NUMBER OF THE PANEL OR SEGMENT CONTACTED
:          -- SEGMENT NUMBERS ARE IN ATB MODEL CODE, NOT BREAK CODE
:       L = 2 IS THE NORMAL FORCE
:       L = 3 IS THE FRICTION FORCE
:       L = 4 TO 6 ARE THE X,Y,Z COORDINATES OF THE CONTACT POINT
:
:       SEGMENT CONTACT POINTS ARE IN ATB LOCAL COORDINATES
:       PANEL CONTACT POINTS ARE IN ATB INERTIAL COORDINATES
:
:  CHEKP (I,J) AND CHEKS (I,J) ARE CONTACT COUNTERS FOR EACH LIMB
:      AND TIME STEP
:    I = TIME STEP
:    J = LIMB OR MEMBER NUMBER
:
:      THE VALUE STORED IN LOCATION CHEKP OR CHEKS (I,J) IS THE NUMBER
:      OF CONTACTS FOR THAT LIMB AND TIME STEP THAT HAD NON-ZERO
:      FORCES.  CONTACTS WITH ZERO FORCES ARE NOT STORED IN THE ARRAY
:      OR WRITTEN TO THE DISK OR PRINTER.
:
:      NP = THE NUMBER OF RECORDS THAT OCCUR BETWEEN PAGE HEADERS
:           ON THE ATB OUTPUT FILE.  THIS MUST BE CHANGED AS THE
:           TIME STEP THEY USE CHANGES.  THIS IS NOT THE SAME BETWEEN
:           EACH HEADER BLOCK.
:  UCOUNT = THE NUMBER OF CONTACT FILES TO BE LOOKED FOR.  THE
:           NUMBER OF FILES FOUND IS COUNTED, NOT THE NUMBER OF
:           INDIVIDUAL CONTACTS.  UCOUNT DOES NOT INCLUDE FILES HAVING
```

```
:          ALL ZERO FORCES OR SEGMENTS THAT ARE NOT OF INTEREST.
:
:  TMAX = MAXIMUM TIME TO BE FOUND.
:
:  FORMAT # 3416 MUST ALSO BE CHANGED TO SKIP THE PROPER NUMBER OF
:     RECORDS IN EACH FILE IF THE FILE IS OF NO INTEREST.
:
:*************************************************************************
      ON ERROR GOTO 255
      DLT = 0.01    ;  NT = 31;    TMAX = 300.0
      IW = 3  ;     IR = 1   ;   IDISK1 = 9 ;    IDISK2 = 8
      WRITE (IW, 100)
      IP = 0  ;   IDT = INT (10000. * DLT)   ; ICOUNT = 0 ; UCOUNT = 9
                DO 200 I = 1, 500
      READ (IDISK1, 103, END=255) IDUM
      IF (IDUM .EQ. 'SEGMENT NO' ) GO TO 210
      IF (IDUM .NE.  'VEHICLE PA') GO TO 200
:
: --------- PANEL VS. SEGMENT CONTACT ---------
:
      READ (IDISK1, 102) IPAN1,  ISEG1,  IPAN2,  ISEG2
      IFLAG1 = IFLAG2 = 0
:
:  CHECK FOR SEGMENT OF INTEREST
:
            DO 110 J = 1, 8
            IF (ISEG1 .NE. ITEST(J) .AND. ISEG2 .NE. ITEST(J)) GO TO 110
            IF (ISEG1 .NE. ITEST(J)) GO TO 105
            JHOLD1 = J  ;  IFLAG1 = 1
105         IF (ISEG2 .NE. ITEST(J)) GO TO 109
            JHOLD2 = J  ;   IFLAG2 = 2
109         IFLAG3 = IFLAG1 + IFLAG2
            IF (IFLAG3 - 3) 110, 112, 110
110         CONTINUE
:
      IF (IFLAG1 + IFLAG2) 111, 111, 112
:
:  SKIP ENTIRE FILE
:
111   READ (IDISK1,3416) DUMB
      GO TO 200
112   READ (IDISK1,113) DUMB
      NUMPAG = NP(1)
      ICOUNT = ICOUNT + 1
            DO 155 J = 1,11
            IF (J .EQ.  1) GO TO 120
            READ (IDISK1, 114) DUMB
120         DO 150 JL= 1, NUMPAG
            READ (IDISK1, 115) T, FN1, FF1, X1, FN2, FF2, X2
            IT = INT (10. * T)
```

```
          ITT = IT / IDT + 1
          IF ((IT/IDT * IDT) .NE. IT) GO TO 150
          IF (IFLAG2)  500, 130, 129
29        IF (FN2 .EQ. 0.0 .AND. FF2 .EQ. 0.0) GO TO 131
          CHEKP (ITT, JHOLD2) = TEMPCT = CHEKP (ITT, JHOLD2) + 1
          PANEL (ITT, JHOLD2, TEMPCT, 1) = FLOAT (IPAN2)
          PANEL (ITT, JHOLD2, TEMPCT, 2) = FN2
          PANEL (ITT, JHOLD2, TEMPCT, 3) = FF2
               DO 132 K4 = 1,  3
32             PANEL (ITT, JHOLD2, TEMPCT, K4+3) = X2 (K4)
31        IF (IFLAG1) 500 ,  140 ,  130
30        IF (FN1 .EQ. 0.0 .AND. FF1 .EQ. 0.0) GO TO 140
          CHEKP(ITT, JHOLD1) = TEMPCT = CHEKP(ITT, JHOLD1) + 1
          PANEL (ITT, JHOLD1, TEMPCT, 1) = FLOAT (IPAN1)
          PANEL (ITT, JHOLD1, TEMPCT, 2) = FN1
          PANEL (ITT, JHOLD1, TEMPCT, 3) = FF1
               DO 133 K4 = 1, 3
133            PANEL (ITT, JHOLD1, TEMPCT, K4+3) = X1 (K4)
140       IF (ITT .GE. NT .OR. T .EQ. TMAX) GO  TO 200
150       CONTINUE
          NUMPAG = NP(J+1)
55        CONTINUE
00             CONTINUE
:
05    READ (IDISK1, 103, END = 255) IDUM
      IF (IDUM  .NE.  'SEGMENT NO') GO TO 260
:
: --------- SEGMENT VS. SEGMENT CONTACT ---------
:
10    BACKSPACE IDISK1
      READ (IDISK1, 101) IDUM, IHOLD1, ISEG1, IHOLD2, ISEG2
      IFLAG1 = IFLAG2 = 0
      JHOLD1 = JHOLD2 = 0
:
:  CHECK FOR SEGMENT(S) OF INTEREST
:
          DO 220 J = 1, 8
          IF (ISEG1 .NE. ITEST(J) .AND. ISEG2 .NE. ITEST(J)) GO TO 220
          IF (ISEG1 .NE. ITEST(J)) GO TO 218
          JHOLD1 = J   ;  IFLAG1 = 1
          GO TO 220
18        IF (ISEG2 .NE. ITEST(J)) GO TO 219
          JHOLD2 = J   ;   IFLAG2 = 2
19        IFLAG3 = IFLAG1 + IFLAG2
          IF (IFLAG3 - 3)  220, 222, 220
20        CONTINUE
:
      IF (IFLAG2 + IFLAG1) 500, 221, 222
:
:  SKIP ENTIRE FILE
```

```
C                                                                       
221   READ (IDISK1,3416) DUMB
      GO TO 260
222   READ (IDISK1,113) DUMB
      NUMPAG = NP(1)
      ICOUNT = ICOUNT + 1
      ITT = 1
            DO 252 J = 1,11
            IF (J .EQ. 1) GO TO 224
            READ (IDISK1,223) DUMB
224         DO 250 JL = 1, NUMPAG
            READ (IDISK1, 225, END=255) T, FN1, FF1, X1, X2
            IF (FN1 .EQ. 0.0 .AND. FF1 .EQ. 0.0) GO TO 240
            IT = INT ( 10. * T )
            ITT = IT / IDT + 1
            IF((IT / IDT * IDT) .NE. IT) GO TO 240
            IF (IFLAG2) 500, 230, 226
226         CHEKS (ITT, JHOLD2) = TEMPCT = CHEKS (ITT, JHOLD2) + 1
            SEGMET (ITT, JHOLD2, TEMPCT, 1) = FLOAT (JHOLD1)
            SEGMET (ITT, JHOLD2, TEMPCT, 2) = FN1
            SEGMET (ITT, JHOLD2, TEMPCT, 3) = FF1
                  DO 227 K4 = 1,  3
227               SEGMET (ITT, JHOLD2, TEMPCT, K4+3) = X2 (K4)
            IF (IFLAG1)  500,  240,  230
230         CHEKS (ITT, JHOLD1) = TEMPCT = CHEKS (ITT, JHOLD1) + 1
            SEGMET (ITT, JHOLD1, TEMPCT, 1) = FLOAT (JHOLD2)
            SEGMET (ITT, JHOLD1, TEMPCT, 2) = FN1
            SEGMET (ITT, JHOLD1, TEMPCT, 3) = FF1
                  DO 232 K4 = 1,  3
232               SEGMET (ITT, JHOLD1, TEMPCT, K4+3) = X1 (K4)
 240        IF (ITT. GE. NT .OR. T .EQ. TMAX) GO TO 260
250         CONTINUE
            NUMPAG = NP(J+1)
252         CONTINUE
C
260   IF (ICOUNT .LT. UCOUNT) GO TO 205
C
C  OUTPUT DATA
C
255         DO 1000 J3 = 1, NT
            TIME = FLOAT (J3-1) * DLT
C2          WRITE (3,2200) TIME
C3          WRITE (IDISK2) TIME
                  DO 1000 J4 = 1, 8
                  ID(1) = J4
                  IF (CHEKP(J3,J4) .EQ. 0) THEN DO
                        JPAN = ID2 = ID(2) = 1
                              ELSE DO
                        JPAN = 6  ; ID(2) = 6 * CHEKP (J3,J4)
                        ID2 = ID(2) / 6
```

```
                    ENDIF
:
                IF (CHEKS(J3,J4) .EQ. 0) THEN DO
                    JSEG = ID3 = ID(3) = 1
                        ELSE DO
                    JSEG = 6  ; ID(3) = 6 * CHEKS(J3,J4)
                    ID3 = ID(3) / 6
                ENDIF
:2    WRITE (3,2000) ITEST(ID(1)), ID(1), ID(2),
:2   $  ((PANEL(J3,J4,J5,J6), J6 = 1, JPAN), J5 = 1, ID2)
:2    WRITE (3,2100) ID(3), ((SEGMET(J3,J4,J5,J6),J6=1,JSEG),J5=1,ID3)
:3        WRITE(IDISK2) ID, ((PANEL(J3,J4,J5,J6),J6=1,JPAN),J5=1,ID2),
:3   *    ((SEGMET(J3,J4,J5,J6),J6=1,JSEG),J5=1,ID3)
1000      CONTINUE
      STOP
500   PRINT,'THE VALUE OF THE FLAG IS L.T. ZERO ---ERROR ---'
      STOP
100   FORMAT (1H1, 10X, 'FORCES FROM ATB MODEL'/)
101   FORMAT (44X,A10,2X,I2,2X,A3,19X,I2,2X,A3)
102   FORMAT (/, 20X, I2, 40X, A3, 15X, I2, 40X, A3)
103   FORMAT (44X,A10)
3416  FORMAT (180(/),A10)
113   FORMAT (///A10)
114   FORMAT (11(/),A10)
115   FORMAT (F9.3,9X,2F9.2,9X,3F8.3,9X,2F9.2,9X,3F8.3)
223   FORMAT (10(/),A10)
225   FORMAT (F9.3,9X,2F9.2,9X,3F8.3,2X,3F8.3)
2000  FORMAT (1H , A4, 2I5, 5X, F5.0, 5F12.3)
 2100 FORMAT (1H , I14, 5X, F5.0, 5F12.3)
 2200 FORMAT ('-TIME =',F6.3,' MSEC')
      END
$DATA
$END
$STOP
/*
//*PW=BONE
//
```

```
'/ALF··.FA JOB DU.D08.AQ0221,BRANDEAU,M=(2,0)
'/  EXEC  WATFIV
'/GO.FT06F001 DD DSN=DU.D08.AQ0221.BRANDEAU.ATBDATA.ONE,DISP=SHR
'/GO.FT07F001 DD DSN=DU.D08.AQ0221.BRANDEAU.ATBDATA.TWO,DISP=SHR
'/GO.FT09F001 DD DSN=DU.D08.AQ0221.BRANDEAU.ATBDATA.FINAL,DISP=SHR
'/GO.SYSIN DD *
;JOB
:$OPTIONS NOEXT,NOCHECK,CCOMP=0
      REAL VEH(13), FORCE(8,30), PANEL(8,24), INSEG(8,24), TIME
      REAL DUMMY(7), OUTSEG(8,72), POSIT(24), CNSTRN(8,3)
      INTEGER NT,  LIMB, ID(8,3), KT(8)
      READ (6) NT, DLT, KCON4, (POSIT(I), I = 1, KCON4)
      WRITE (9) NT, DLT, KCON4, (POSIT(I), I = 1, KCON4)
:
:  READ AND WRITE SEGMENT #, MASS, INERTIA, AND SEMI-MAJOR AXES.
:
      DO 10 I = 1, 8
        READ (6) J, DUMMY
   10 WRITE (9) J, DUMMY
:
:  BEGIN LOOP FOR ALL REMAINING DATA TO BE COMBINED
:
      DO 100 I = 1, NT
      READ (6) VEH
      READ (7) TIME
      PRINT,'TIME = ',TIME
      WRITE (9) VEH
:
:  READ IN ALL DATA FOR THIS TIME STEP
:
      DO 40 J = 1, 8
        READ (6) LIMB, K3, (FORCE(J,K), K =1,30),(CNSTRN(J,K),K=1,K3)
        KT(J) = K3
:
:  CONVERT ROTATIONS FROM DEGREES TO RADIANS
:
      DO 20 K = 7,9
   20 FORCE(J,K) = FORCE(J,K) / 57.29578
      READ (7) ID(J,1), ID2, ID3, (PANEL(J,KK), KK = 1, ID2),
     $  (INSEG(J,KK), KK = 1, ID3)
      ID(J,2) = ID2 ;  ID(J,3) = ID3
      IF (LIMB .NE. ID(J,1)) THEN DO
        PRINT,'LIMB NOS. NOT EQUAL AT STEP ',J
        PRINT,'LIMB NO. FROM UNIT 6 =',LIMB
        PRINT,'LIMB NO. FROM UNIT 7 =',ID(J,1)
        PRINT,'STOPPING NOW'
        STOP
      ENDIF
   40 CONTINUE
:
```

```
:  COMBINE DATA FOR EACH CONTACTED SEGMENT SO THAT RELATIVE
:  VELOCITIES CAN BE CALCULATED.  INCREASE SEGMENT DATA FROM
:  6 TO 18 ITEMS.
:
      DO 60 J = 1, 8
        ID3 = ID(J,3) ;  ID2 = ID(J,2);  OUTSEG(J,1) = 0.0
        IF (ID3 .EQ. 1) GO TO 55
        DO 50 K = 1, ID3, 6
          NUM = IFIX(INSEG(J,K))
          K1 = 3 * K - 2 ; K2 = K1 + 17 ; K3 = K1 + 6 ; K4 = K1 + 5
          DO 44 L = K1, K4
   44     OUTSEG(J,L) = INSEG(J,K+L-K1)
          IF (NUM .EQ. 0) THEN DO
            DO 45 L = K3, K4
   45       OUTSEG(J,L) = 0.0
          ELSE DO
:
:  FIND PROPER LOCATION IN SEGMENT "NUM" FILE
:
            ID2 = 1
            WHILE (IFIX(INSEG(NUM,ID2)) .NE. J) DO
              ID2 = ID2 + 6
            ENDWHILE
:
            DO 48 L = 1, 3
   48       OUTSEG(J,K1+L+5) = FORCE(NUM,L+3)
            DO 49 L = 15, 17
   49       OUTSEG(J,K1+L) = INSEG(NUM,ID2+L-12)
          ENDIF
   50   CONTINUE
        ID3 = ID3 / 6 * 18
:
:  NEGATE "OTHER FORCES" FOR EQUILIBRIUM
:
   55   K3 = KT(J)
        WRITE (9) ID(J,1), ID2, ID3, K3, (FORCE(J,KK), KK = 1, 30),
     $   (PANEL(J,KK),KK = 1, ID2), (OUTSEG(J,KK), KK = 1, ID3)
     $   ,(-CNSTRN(J,KK), KK = 1, K3)
:2      WRITE (3,300) ID(J,1), ID2, (PANEL(J,KK), KK = 1, ID2)
:2      WRITE (3,200) ID3, (OUTSEG(J,KK), KK = 1, ID3)
:2      IF (K3 .GT. 1) WRITE (3,150) K3, (-CNSTRN(J,KK), KK = 1,K3)
   60 CONTINUE
  100 CONTINUE
      STOP
:2300 FORMAT (' LIMB =',I2,I3,5X,(2(F5.0,5F8.2,5X)))
:2200 FORMAT (' SEGS',I3,(F5.0,1X, 2F6.2, 2X, 15F7.1))
:2150 FORMAT ('EXTRA CONTACTS =', I4, 3F12.4)
      END
:DATA
:END
```

```
:SOPTIONS DEC,CCOMP=1
:  PROGRAM NAME : CONTACT
      REAL VEH(12), FORCE(30), PANEL(24), SEGMET(72), PLANE(33)
      REAL ABC(24), XYZ(24), TIME, INERT(24), WEIGHT(8), D1(8)
      REAL RADS(3), OMEGA(3), THOLD(3), OMEGA2(3), RADS2(3)
      REAL XHOLD(3), XTEMP(3), TEMP(3), WORK(6), POSIT(24), CNSTRN(6)
      CHARACTER SEG*3(8) /'RUL','RLL','LUL','LLL','RUA','RLA','LUA',
     *      'LLA' /
      INTEGER NT, LIMB, NPAN, NSEG, KT, JS, JS2, JS3, LIMB0, LIMB1, KKK
      INTEGER KK, J1, J2, J, I, K, ICHEK, KS, KP, NUM, NUM0, NCON, KCON4
      COMMON /TRANS/ ICHEK
      INTEGER YES(8)
      DATA YES /8 * 0 /
      REAL GINCH / 386.0886 /
      DATA PLANE /-.1104,0.0,-.9939,     .9744,0.0,-.2249,
     *   0.0,0.0,-1.0,       .9745,0.0,-.2245,    .9191,0.0,.3939,
     *   -.0499,0.0,.9988,      -1.0,0.0,0.0,       0.0,-1.0,0.0,
     *   0.0,1.0,0.0,    .9720,0.0,-.2350,    -.6428,0.0,-.7660 /
      DATA D1 /8.8, 7.93, 8.8, 7.93, 5.44, 7.94, 5.44, 7.94 /
      EQUIVALENCE (RADS, FORCE(7)), (OMEGA,FORCE(10))
      READ (5) NT, DLT, KCON4, (POSIT(I), I = 1, KCON4)
      IF (KCON4 .EQ. 1) GO TO 4
:
:  ROTATE "OTHER CONSTRAINT FORCE" LOCATION TO X-AXIAL COORDINATES
:  WITH THE ORIGIN AT THE PROXIMAL JOINT
:
      DO 3 I = 1, KCON4, 4
        CALL CHANGE (POSIT, I+1, 24, D1(POSIT(I)))
    3 YES(POSIT(I)) = I + 1
:
    4 WRITE (9) NT, DLT, KCON4, (POSIT(I), I = 1, KCON4)
      WRITE (9) D1(1), D1(2), D1(5), D1(6)
:
:  READ SEGMENT WEIGHTS, INERTIAS, AND SEMI-MAJOR AXES IN LOCAL X-AXIAL
:    SEGMENT COORDINATES.
:
      J1 = -2
      DO 5 J = 1, 8
      J1 = J1 + 3
      J2 = J1 + 2
      READ (5) I,WEIGHT(J), (INERT(K), K=J1,J2), (ABC(K), K = J1, J2)
      WRITE (9) I, WEIGHT(J), (ABC(K), K=J1,J2)
:
:  SQUARE SEMI-MAJOR AXES OF ELLIPSOIDS FOR LATER USE
:
      DO 5 K = J1, J2
    5 ABC(K) = ABC(K) * ABC(K)
:
      DO 350 KKK = 1, NT
      READ (5) TIME, VEH
```

```
C1    PRINT,'TIME =',TIME
      WRITE (9) TIME
      DO 350 KK = 1, 8
      READ (5) LIMB, NPAN, NSEG, NCON, FORCE, (PANEL(I), I = 1, NPAN)
     *   ,(SEGMET(I), I = 1,NSEG), (CNSTRN(I), I = 1, NCON)
      LIMBO = 3 * LIMB - 3
C
C3    PRINT,'**************************************************'
C3    PRINT,'SEGMENT IS #',LIMB,' OR ',SEG(LIMB)
C3    PRINT,'**************************************************'
C3    PRINT,'ROTATION ANGLES (RADIANS)'
C3    WRITE (3,500) (FORCF(I),I = 7, 9)
      ICHEK = -1
      KS = 0
      KP = 0
      IF (NSEG .EQ. 1) GO TO 50
C---------------------------------------------------------------------
C                    SEGMENT - SEGMENT CONTACT
C---------------------------------------------------------------------
C  SEGMENT CONTACT POINTS IN SEGMENT COORDINATES
C
      KS = NSEG / 18
      DO 40 J = 1, KS
      JXYZ = 6 * J - 5
      JS = 1 + (J-1) * 18
      JS2 = JS + 1
      JS3 = JS + 2
      NUM = IFIX (SEGMET(JS))
C3    PRINT,' '
C3    PRINT,'SEGMENT',LIMB,' CONTACTING SEGMENT',NUM
C3    PRINT,'NORMAL FORCE =',SEGMET(JS2)
C3    PRINT,'FRICTION FORCE =', SEGMET(JS3)
C3    PRINT,'CONTACT AT ATB LOCAL XYZ',(SEGMET(JS+I), I = 3, 5)
C            ------------------------
C              NORMAL CONTACT FORCES
C            ------------------------
C
C  FIND NORMAL TO ELLIPSOID AT POINT OF CONTACT
C
      HOLD = 0.0
      DO 10 I = 1, 3
        HOLD2 = SEGMET(JS3+I) / ABC(LIMBO+I)
        TEMP(I) = HOLD2
        WORK(I+3) = HOLD2
   10 HOLD = HOLD + HOLD2 * HOLD2
      HOLD = SQRT (HOLD)
C
C  MULTIPLY UNIT NORMAL BY NCRMAL FORCE SCALAR - STORE IN "WORK(1-3)"
C
      DO 15 I = 1, 3
```

```
   15 WORK(I) = -TEMP(I) * SEGMET(JS2) / HOLD
      IF (SEGMET(JS3) .EQ. 0.0) GO TO 39
C              ----------------------
C                FRICTION CONTACT FORCES
C              ----------------------
C
C     ASSIGN DATA FOR SEGMENT "NUM" TO STORAGE VECTORS
C
      DO 20 I = 1, 3
        JSI = JS + I
        XTEMP(I) = SEGMET(JSI + 14)
        THOLD(I) = SEGMET(JSI + 5)
        RADS2(I) = SEGMET(JSI + 8)
        OMEGA2(I) = SEGMET(JSI + 11)
   20 TEMP(I) = SEGMET(JSI + 2)
C
C     CONSTRUCT RELATIVE VELOCITY VECTOR IN SEGMENT "LIMB" COORDINATES
C     FOR SEGMENT "LIMB" CONTACTING SEGMENT "NUM".
C     VECTORS "OMEGA2", "RADS2", "THOLD" & "XTEMP" CONTAIN DATA FOR
C     SEGMENT "NUM".
C
      CALL CROSS (OMEGA2, XTEMP, XHOLD)
C9    PRINT,'W2 CROSS R2=',XHOLD
      CALL ROT (XHOLD, 1, 3, RADS2, -1)
C9    PRINT,'SAME VECTOR ROTATED INTO INERTIAL',XHOLD
      ICHEK = -1
      CALL CROSS (OMEGA, TEMP, XTEMP)
C9    PRINT,'W CROSS R=',XTEMP
      DO 25 I = 1, 3
   25 THOLD(I) = FORCE(I+3) - THOLD(I) - XHOLD(I)
      CALL ROT (THOLD, 1, 3, RADS, +1)
C
      DO 30 I = 1, 3
        THOLD(I) = THOLD(I) + XTEMP(I)
   30 TEMP(I) = WORK(I+3)
C9    PRINT,'RELATIVE VELOCITY VECTOR IN LOCAL',THOLD
C
C     PROJECT VELOCITY VECTOR ONTO TANGENT PLANE TO SEGMENT "LIMB"
C
      CALL VECTOR (TEMP, THOLD, XHOLD)
C9    PRINT,'SAME PROJECTED INTO PLANE OF SEG.',XHOLD
C
C     MULTIPLY PROJECTED VELOCITY BY FRICTION SCALAR AND COMBINE
C     ALL FORCES INTO "WORK (1-3)"
C
      DO 35 I = 1, 3
   35 WORK(I) = WORK(I) + XHOLD(I) * SEGMET(JS3)
C     CONVERT SEGMENT Z-AXIAL COORDINATES TO X-AXIAL
C
   39 XYZ(JXYZ) = SEGMET(JS+5) + D1(LIMB)
```

```
      XYZ(JXYZ+1) = SEGMET(JS+3)
      XYZ(JXYZ+2) = SEGMET(JS+4)
      XYZ(JXYZ+3) = WORK(3)
      XYZ(JXYZ+4) = WORK(1)
      XYZ(JXYZ+5) = WORK(2)
   40 CONTINUE
   50 IF (NPAN .EQ. 1) GO TO 130
C-----------------------------------------------------------------------
C                          SEGMENT - PANEL CONTACT
C-----------------------------------------------------------------------
C  PANEL CONTACT DATA IS IN INERTIAL COORDINATES.
C
      KP = NPAN / 6
      K1 = KS + 1
      K2 = K1 + KP - 1
      DO 120 J1 = K1, K2
      JXYZ = 6 * J1 - 5
      J = J1 - KS
      JS = 1 + (J-1) * 6
      JS2 = JS + 1
      JS3 = JS + 2
C
C  CORRECT PANEL CONTACT POINT FOR VEHICLE MOTION
C
      DO 55 I2 = 1, KP
        DO 55 I = 1, 3
        NUM = 6 * (J-1) + I + 3
   55 PANEL(NUM) = PANEL(NUM) + VEH(I)
C3    NUM = IFIX(PANEL(JS))
C3    PRINT,' '
C3    PRINT,'SEGMENT',LIMB,' CONTACTING PANEL',NUM
C3    PRINT,'NORMAL FORCE =',PANEL(JS2)
C3    PRINT,'FRICTION FORCE =',PANEL(JS3)
C3    PRINT,'CONTACT AT INERTIAL XYZ',(PANEL(JS+I), I = 3, 5)
C
C  FIND VECTOR FROM CENTER OF ELLIPSOID TO CONTACT POINT
C  IN INERTIAL COORDINATES.
C
      DO 60 I = 1, 3
        TEMP(I) = PANEL(JS3+I) - FORCE(I)
   60 THOLD(I) = TEMP(I)
C5    PRINT,'INERTIAL CONTACT VECTOR BEFORE TRANSFORMATION'
C5    WRITE (3,500) TEMP
C
C  TRANSFORM CONTACT VECTOR FROM INERTIAL TO SEGMENT COORDINATES
C
      CALL ROT (TEMP, 1, 3, RADS, 1)
C5    PRINT,'AFTER TRANSFORMATION TO SEGMENT LOCAL'
C5    WRITE (3,500) TEMP
C5    PRINT,' '
```

```
C
C   STORE CONTACT POINT IN X-AXIAL COORDINATES
C
      XYZ(JXYZ) = TEMP(3) + D1(LIMB)
      XYZ(JXYZ+1) = TEMP(1)
      XYZ(JXYZ+2) = TEMP(2)
C             ------------------------
C                 NORMAL CONTACT FORCES
C             ------------------------
      NUM = IFIX (PANEL(JS))
      NUMO = 3 * NUM - 3
      DO 70 I = 1, 3
        TEMP(I) = PLANE(NUMO+I)
        WORK(I+3) = 0.0
        XTEMP(I) = OMEGA(I)
   70 WORK(I) = TEMP(I) * PANEL(JS2)
C
      IF (PANEL(JS3) .EQ. 0.0) GO TO 95
C             ------------------------
C                FRICTION CONTACT FORCES
C             ------------------------
C
C   FIND VELOCITY VECTOR OF CONTACT POINT IN INERTIAL COORDINATES.
C
      CALL ROT2 (XTEMP, 1, 3, -1)
      CALL CROSS (XTEMP, THOLD, XHOLD)
C6    PRINT,'ANGULAR VELOCITY OF C.G.'
C6    WRITE (3,500) XTEMP
C6    PRINT,'LINEAR VELOCITY DUE TO ROTATION -- OMEGA CROSS R'
C6    WRITE (3,500) XHOLD
C6    PRINT,'TRANSLATIONAL VELOCITY'
C6    WRITE (3,500) (FORCE(I), I = 4, 6)
      DO 80 I = 1, 3
   80 XHOLD(I) = XHOLD(I) + FORCE(I+3)
C6    PRINT,'TOTAL VELOCITY VECTOR'
C6    WRITE (3,500) XHOLD
C
C  PROJECT VELOCITY VECTOR ONTO TANGENT PLANE
C
      CALL VECTOR (TEMP, XHOLD, THOLD)
C6    PRINT,'UNIT VECTOR OF VELOCITY PROJECTED ONTO TANGENT PLANE'
C6    WRITE (3,500) THOLD
C
C   MULTIPLY NEG. UNIT VECTOR BY FRICTION SCALAR TO GET FRICTION VECTOR
C
      DO 90 I = 1, 3
   90 WORK(I+3) = -THOLD(I) * PANEL(JS3)
C
C   CREATE TOTAL FORCE VECTOR IN INERTIAL COORDINATES
C
```

```
   95 DO 100 I = 1, 3
  100 TEMP(I) = WORK(I) + WORK(I+3)
C6    PRINT,'TOTAL FORCE VECTOR IN INERTIAL COORDINATES'
C6    WRITE (3,500) TEMP
C
C  TRANSFORM TOTAL FORCE VECTOR TO SEGMENT Z-AXIAL COORDINATES.
C
      CALL ROT2 (TEMP, 1, 3, 1)
C5    PRINT,'AFTER CHANGING TO X-AXIAL COORDINATES'
C5    WRITE (3,500) TEMP
C5    PRINT,' '
C
C  CONVERT SEGMENT Z-AXIAL COORDINATES TO X-AXIAL.
C
      XYZ(JXYZ+3) = TEMP(3)
      XYZ(JXYZ+4) = TEMP(1)
      XYZ(JXYZ+5) = TEMP(2)
  120 CONTINUE
C----------------------------------------------------------------------
C  JOINT FORCES & TORQUES; LINEAR & ANGULAR ACCELERATIONS
C----------------------------------------------------------------------
C  IN BOTH INERTIAL AND LOCAL COORDINATES
C     LOCAL : ANGULAR VELOCITIES, ACCELERATIONS, U2 ARRAY
C INERTIAL : JOINT FORCES & TORQUES, LINEAR ACCEL., U1 ARRAY
C
  130 KT = KS + KP
      KT6 = KT * 6
C
C  TRANSFORM INERTIAL VECTORS TO SEGMENT X-AXIAL LOCAL VECTORS.
C
C4    PRINT,'W,FORCES, TORQUES, ALPHA, ACCEL., U1, U2 IN ORIG. COORDS'
C4    WRITE (3,600) (FORCE(I),I=10,30)
      DO 134 J = 13, 25, 3
        IF (J .EQ. 19) GO TO 134
        CALL ROT (FORCE, J, 30, RADS, +1)
        CALL CHANGE (FORCE, J, 30, 0.0)
  134 CONTINUE
C
C  ROTATE LOCAL VECTORS TO X-AXIAL
C
      DO 136 J = 10, 28, 9
  136 CALL CHANGE (FORCE, J, 30, 0.0)
C4    PRINT,'SAME ITEMS IN SEGMENT X-AXIAL COORDINATES'
C4    WRITE (3,600) (FORCE(J), J = 10, 30)
      IF (YES(LIMB)) 140, 140, 138
C
C  TRANSFORM "OTHER CONSTRAINT FORCE" TO SEGMENT LOCAL X-AXIAL
C
  138 CALL ROT (CNSTRN, 1, 6, RADS, +1)
      CALL CHANGE (CNSTRN, 1, 6, 0.0)
```

```
  140 KS = YES(LIMB)
      IF (KT .LT. 1 .AND. KS .LT. 1) GO TO 150

   CORRECT U1 & U2 ARRAYS TO ALLOW FOR POINT FORCE APPLICATIONS.

      DO 142 I = 1, 3
  142 XHOLD(I) = 0.0
      IF (KS .LT. 1) GO TO 144

   SUM TORQUES ABOUT C.G. DUE TO OTHER CONSTRAINT FORCES.

      XHOLD(1) = POSIT(KS+1) * CNSTRN(3) - POSIT(KS+2) * CNSTRN(2)
      XHOLD(2) = -(POSIT(KS)-D1(LIMB))*CNSTRN(3) + POSIT(KS+2)*CNSTRN(1)
      XHOLD(3) = (POSIT(KS)-D1(LIMB))*CNSTRN(2) - POSIT(KS+1)*CNSTRN(1)
      IF (KT .LE. 1) GO TO 146

   SUM TORQUES ABOUT C.G. DUE TO CONTACT FORCES.

  144 GMASS = GINCH / WEIGHT(LIMB)
      DO 145 J = 1, KT
      I = 6 * J - 5
      XHOLD(1) = XHOLD(1) + XYZ(I+1) * XYZ(I+5)
     $             - XYZ(I+2) * XYZ(I+4)
      XHOLD(2) = XHOLD(2) - (XYZ(I)-D1(LIMB)) * XYZ(I+5)
     $             + XYZ(I+2) * XYZ(I+3)
      XHOLD(3) = XHOLD(3) + (XYZ(I)-D1(LIMB)) * XYZ(I+4)
     $             - XYZ(I+1) * XYZ(I+3)
      DO 145 JJ = 3, 5
      FORCE(JJ+22) = FORCE(JJ+22) - GMASS * XYZ(I+JJ)
  145 CONTINUE
  146 CONTINUE
8     PRINT,'U2 ARRAY IN X-AXIAL COORDINATES'
8     WRITE (3,500) (FORCE(J), J = 28,30)
8     PRINT,'TORQUES DUE TO ALL POINT SEGMENT FORCES ABOUT C.G.'
8     WRITE(3,500) XHOLD

   ADD T/I TO U2 TO PRODUCE ADJUSTED ACCELERATION.

      DO 148 J = 1, 3
  148 FORCE(J+27) = FORCE(J+27) + XHOLD(J) / INERT(J+LIMB0)
8     PRINT,'ADJUSTED U2'
8     WRITE (3,500) (FORCE(J), J = 28,30)

   COMBINE A AND U1 INTO LOCATION OF A.

  150 DO 152 J = 22, 24
  152 FORCE(J) = FORCE(J) - FORCE(J+3) / GINCH

   ORDER CONTACTS BY INCREASING X VALUE OF CONTACTING POINT
```

```
      IF (KT .LE. 1) GO TO 200
      KT1 = KT -1
      DO 170 I = 1, KT1
        J1 = 6 * I - 5
        HOLD = XYZ(J1)
        I1 = I + 1
        NUM = J1
        DO 158 J = I1, KT
          J2 = 6 * J - 5
          IF (XYZ(J2) .GE. HOLD) GO TO 158
          NUM = J2
          HOLD = XYZ(J2)
  158   CONTINUE
        IF (NUM .EQ. J1) GO TO 170
        J1 = J1 - 1
        NUM = NUM - 1
        DO 160 K = 1, 6
          J3 = 6 * K - 5
          HOLD = XYZ(J1+K)
          XYZ(J1+K) = XYZ(NUM+K)
  160   XYZ(NUM+K) = HOLD
  170 CONTINUE
  200 IF (KT) 205, 205, 210
  205 KT = 1
      XYZ(1) = 0.0
      GO TO 300
  210 KT = KT * 6
  300 CONTINUE
C4    PRINT,'CONTACT FORCES'
C4    PRINT,'NUMBER OF DATA ITEMS =',KT
C4    K2 = KT / 6
C4    IF (KT .GT. 1) WRITE (3,600) (XYZ(I), I = 1, KT)
C
C  OUTPUT RESULTS TO DISK
C
      WRITE (9) LIMB,KT,NCON,(FORCE(I),I = 10,24),(FORCE(I),I = 28,30),
     $       (XYZ(I),I = 1, KT), (CNSTRN(I), I = 1, NCON)
  350 CONTINUE
      STOP
  500 FORMAT(1H ,3F10.4)
  600 FORMAT(1H ,3F10.4,' --- ',3F10.4)
      END
      SUBROUTINE ROT (V, IZ, N, P, K)
C
C  SUBROUTINE TO TRANSFORM VECTOR V THROUGH P RADIANS INTO ANOTHER
C  COORDINATE SYSTEM.  K DETERMINES WHETHER THE TRANSFORMATION IS FROM
C  INERTIAL TO SEGMENT OR VICE-VERSA :
C     K = 1     INERTIAL TO SEGMENT TRANSFORMATION
C     K =-1     SEGMENT TO INERTIAL       "
C
```

```
      REAL R(3,3), V(N), VN(3), P(3)
      COMMON /TRANS/ ICHEK
      IF (ICHEK . EQ. 1) GO TO 10
      ICHEK = 1
      C1 = COS (P(1))
      C2 = COS (P(2))
      C3 = COS (P(3))
      S1 = SIN (P(1))
      S2 = SIN (P(2))
      S3 = SIN (P(3))
      R(1,1) = C2 * C1
      R(2,1) = S3 * S2 * C1 - S1 * C3
      R(3,1) = C3 * S2 * C1 + S1 * S3
      R(1,2) = C2 * S1
      R(2,2) = S3 * S2 * S1 + C1 * C3
      R(3,2) = C3 * S2 * S1 - C1 * S3
      R(1,3) = -S2
      R(2,3) = S3 * C2
      R(3,3) = C3 * C2
      ENTRY ROT2 (V, IZ, N, K)
   10 IZ1 = IZ - 1
      IF (K) 40, 60, 20
   20 DO 30 I = 1, 3
        VN(I) = 0.0
        DO 30 J = 1, 3
   30 VN(I) = VN(I) + R(I,J) * V(IZ1+J)
      GO TO 60
   40 DO 50 I = 1, 3
        VN(I) = 0.0
        DO 50 J = 1, 3
   50 VN(I) = VN(I) + R(J,I) * V(J)
   60 DO 70 I = 1, 3
   70 V(IZ1+I) = VN(I)
      RETURN
      END
      SUBROUTINE VECTOR ( N, V, C)
C
C     SUBROUTINE TO FIND THE VECTOR PROJECTION OF VECTOR V ONTO THE
C     PLANE WHICH N IS NORMAL TO.  THE CROSS PRODUCT IS TAKEN TWICE
C     AND THE RESULTANT VECTOR IS NORMALIZED TO A UNIT VECTOR.
C
      REAL N(3), V(3), C(3), D(3), HOLD
      HOLD = 0.0
      CALL CROSS (N, V, C)
      CALL CROSS (C, N, D)
      DO 10 J = 1, 3
   10 HOLD = HOLD + D(J) * D(J)
      HOLD = SQRT (HOLD)
      IF (HOLD .EQ. 0.0) RETURN
      DO 20 J = 1, 3
```

```
   20 C(J) = D(J) / HOLD
      RETURN
      END
      SUBROUTINE CROSS ( A, B, C )
C
C  SUBROUTINE TO TAKE THE CROSS PRODUCT OF A CROSS B, AND RETURN
C  IT IN VECTOR C.  A ,B AND C ARE ALL VECTORS OF LENGTH 3.
C
      REAL A(3), B(3), C(3)
      C(1) = A(2) * B(3)  -  A(3) * B(2)
      C(2) =-A(1) * B(3)  +  A(3) * B(1)
      C(3) = A(1) * B(2)  -  A(2) * B(1)
      RETURN
      END
      SUBROUTINE CHANGE (TEMP, I, N, D)
C
C  CHANGES Z-AXIAL TO X-AXIAL COORDINATES.  SHIFTS NEW X BY
C  D TO ALLOW TRANSFER OF ORIGIN TO PROXIMAL JOINT.
C
      REAL TEMP(N)
      HOLD1 = TEMP(I)
      HOLD2 = TEMP(I+1)
      HOLD3 = TEMP(I+2)
      TEMP(I) = HOLD3 + D
      TEMP(I+1) = HOLD1
      TEMP(I+2) = HOLD2
      RETURN
      END
      SUBROUTINE FOR (K)
      DO 10 J = 1, K
   10 READ (5)
      RETURN
      END
      SUBROUTINE BACK (K)
      DO 10 J = 1, K
   10 BACKSPACE 5
      RETURN
      END
```

```
 FINDS AREA OF BONE AT VARIOUS LENGTHS
    INTEGER TITLE(18)
    REAL A(40), D(40), DNORM(40), X(40), Y(40), Z(40), U, TEMP
    READ (1,100) TITLE
    WRITE (3,150) TITLE
    PRINT,'NORM LENGTH,  LENGTH,  AREA'
    PRINT,' '
    K = 1
 10 READ (1,*,END=20) D(K), A(K)
    K = K + 1
    GO TO 10
 20 K = K - 1
    DO 30 I = 1, K
      DNORM(I) = D(I) / D(K)
 30 WRITE (3,200) DNORM(I), D(I), A(I)
    CALL SPLINE (K, DNORM, A, X, Y, Z)

 EVALUATE @ 11 POINTS NORMALIZED

    PRINT,' '
    PRINT,' '
    PRINT,'NORM LENGTH & AREA EVALUATED BY SEVAL'
    PRINT,' '
    DO 40 I = 1, 11
      U = FLOAT(I-1) / 10.0
      TEMP = SEVAL (K, U, DNORM, A, X, Y, Z)
 40 WRITE (3,200) U, TEMP
    STOP
100 FORMAT (18A4)
150 FORMAT (' ID = ',18A4)
200 FORMAT (3F10.3)
    END
    SUBROUTINE SPLINE (N, X, Y, B, C, D)
    INTEGER N
    REAL X(N), Y(N), B(N), C(N), D(N)
    INTEGER NM1, IB, I
    REAL T
    NM1 = N-1
    IF ( N .LT. 2 ) RETURN
    IF ( N .LT. 3 ) GO TO 50
    D(1) = X(2) - X(1)
    C(2) = (Y(2) - Y(1))/D(1)
    DO 10 I = 2, NM1
       D(I) = X(I+1) - X(I)
       B(I) = 2.*(D(I-1) + D(I))
       C(I+1) = (Y(I+1) - Y(I))/D(I)
       C(I) = C(I+1) - C(I)
 10 CONTINUE
    B(1) = -D(1)
    B(N) = -D(N-1)
```

```
      C(1) = 0.
      C(N) = 0.
      IF ( N .EQ. 3 ) GO TO 15
      C(1) = C(3)/(X(4)-X(2)) - C(2)/(X(3)-X(1))
      C(N) = C(N-1)/(X(N)-X(N-2)) - C(N-2)/(X(N-1)-X(N-3))
      C(1) = C(1)*D(1)**2/(X(4)-X(1))
      C(N) = -C(N)*D(N-1)**2/(X(N)-X(N-3))
   15 DO 20 I = 2, N
         T = D(I-1)/B(I-1)
         B(I) = B(I) - T*D(I-1)
         C(I) = C(I) - T*C(I-1)
   20 CONTINUE
      C(N) = C(N)/B(N)
      DO 30 IB = 1, NM1
         I = N-IB
         C(I) = (C(I) - D(I)*C(I+1))/B(I)
   30 CONTINUE
      B(N) = (Y(N) - Y(NM1))/D(NM1) + D(NM1)*(C(NM1) + 2.*C(N))
      DO 40 I = 1, NM1
         B(I) = (Y(I+1) - Y(I))/D(I) - D(I)*(C(I+1) + 2.*C(I))
         D(I) = (C(I+1) - C(I))/D(I)
         C(I) = 3.*C(I)
   40 CONTINUE
      C(N) = 3.*C(N)
      D(N) = D(N-1)
      RETURN
   50 B(1) = (Y(2)-Y(1))/(X(2)-X(1))
      C(1) = 0.
      D(1) = 0.
      B(2) = B(1)
      C(2) = 0.
      D(2) = 0.
      RETURN
      END
      REAL FUNCTION SEVAL(N, U, X, Y, B, C, D)
      INTEGER N
      REAL  U, X(N), Y(N), B(N), C(N), D(N)
      INTEGER I, J, K
      REAL DX
      DATA I/1/
      IF ( I .GE. N ) I = 1
      IF ( U .LT. X(I) ) GO TO 10
      IF ( U .LE. X(I+1) ) GO TO 30
   10 I = 1
      J = N+1
   20 K = (I+J)/2
      IF ( U .LT. X(K) ) J = K
      IF ( U .GE. X(K) ) I = K
      IF ( J .GT. I+1 ) GO TO 20
   30 DX = U - X(I)
```

```
      SEVAL = Y(I) + DX*(B(I) + DX*(C(I) + DX*D(I)))
      RETURN
      END
$DATA
FEMUR #1 -- RIGHT   DISTAL VS. AVG. INERTIA
0.0    .226
1.0    .092
2.0    .076
3.0    .075
4.0    .074
5.0    .069
6.2    .067
7.2    .065
8.2    .066
9.2    .067
10.2   .073
11.2   .085
12.2   .105
13.2   .249
14.2   1.818
```

```
      DOUBLE PRECISION FUNCTION ZEROIN(AX,BX,F,TOL)
      DOUBLE PRECISION AX,BX,F,TOL
C
C      A ZERO OF THE FUNCTION  F(X)  IS COMPUTED IN THE INTERVAL AX,BX .
C
C  INPUT..
C
C  AX     LEFT ENDPOINT OF INITIAL INTERVAL
C  BX     RIGHT ENDPOINT OF INITIAL INTERVAL
C  F      FUNCTION SUBPROGRAM WHICH EVALUATES F(X) FOR ANY X IN
C         THE INTERVAL  AX,BX
C  TOL    DESIRED LENGTH OF THE INTERVAL OF UNCERTAINTY OF THE
C         FINAL RESULT ( .GE. 0.0D0)
C
C
C  OUTPUT..
C
C  ZEROIN ABCISSA APPROXIMATING A ZERO OF  F  IN THE INTERVAL AX,BX
C
C
C      IT IS ASSUMED THAT  F(AX)  AND  F(BX)  HAVE OPPOSITE SIGNS
C  WITHOUT A CHECK.  ZEROIN RETURNS A ZERO  X  IN THE GIVEN INTERVAL
C  AX,BX  TO WITHIN A TOLERANCE  4*MACHEPS*ABS(X) + TOL, WHERE MACHEPS
C  IS THE RELATIVE MACHINE PRECISION.
C      THIS FUNCTION SUBPROGRAM IS A SLIGHTLY  MODIFIED  TRANSLATION  OF
C  THE ALGOL 60 PROCEDURE  ZERO  GIVEN IN  RICHARD BRENT, ALGORITHMS FOR
C  MINIMIZATION WITHOUT DERIVATIVES, PRENTICE - HALL, INC. (1973).
C
C
      DOUBLE PRECISION  A,B,C,D,E,EPS,FA,FB,FC,TOL1,XM,P,Q,R,S
      DOUBLE PRECISION  DABS,DSIGN
C
C  COMPUTE EPS, THE RELATIVE MACHINE PRECISION
C
      EPS = 1.0D0
   10 EPS = EPS/2.0D0
      TOL1 = 1.0D0 + EPS
      IF (TOL1 .GT. 1.0D0) GO TO 10
C
C INITIALIZATION
C
      A = AX
      B = BX
      FA = F(A)
      FB = F(B)
C
C BEGIN STEP
C
   20 C = A
      FC = FA
```

```
      D = B - A
      E = D
   30 IF (DABS(FC) .GE. DABS(FB)) GO TO 40
      A = B
      B = C
      C = A
      FA = FB
      FB = FC
      FC = FA
C
C CONVERGENCE TEST
C
   40 TOL1 = 2.0D0*EPS*DABS(B) + 0.5D0*TOL
      XM = .5*(C - B)
      IF (DABS(XM) .LE. TOL1) GO TO 90
      IF (FB .EQ. 0.0D0) GO TO 90
C
C IS BISECTION NECESSARY
C
      IF (DABS(E) .LT. TOL1) GO TO 70
      IF (DABS(FA) .LE. DABS(FB)) GO TO 70
C
C IS QUADRATIC INTERPOLATION POSSIBLE
C
      IF (A .NE. C) GO TO 50
C
C LINEAR INTERPOLATION
C
      S = FB/FA
      P = 2.0D0*XM*S
      Q = 1.0D0 - S
      GO TO 60
C
C INVERSE QUADRATIC INTERPOLATION
C
   50 Q = FA/FC
      R = FB/FC
      S = FB/FA
      P = S*(2.0D0*XM*Q*(Q - R) - (B - A)*(R - 1.0D0))
      Q = (Q - 1.0D0)*(R - 1.0D0)*(S - 1.0D0)
C
C ADJUST SIGNS
C
   60 IF (P .GT. 0.0D0) Q = -Q
      P = DABS(P)
C
C IS INTERPOLATION ACCEPTABLE
C
      IF ((2.0D0*P) .GE. (3.0D0*XM*Q - DABS(TOL1*Q))) GO TO 70
      IF (P .GE. DABS(0.5D0*E*Q)) GO TO 70
```

```
      E = D
      D = P/Q
      GO TO 80
:
: BISECTION
:
   70 D = XM
      E = D
:
: COMPLETE STEP
:
   80 A = B
      FA = FB
      IF (DABS(D) .GT. TOL1) B = B + D
      IF (DABS(D) .LE. TOL1) B = B + DSIGN(TOL1, XM)
      FB = F(B)
      IF ((FB*(FC/DABS(FC))) .GT. 0.0D0) GO TO 20
      GO TO 30
:
: DONE
:
   90 ZEROIN = B
      RETURN
      END
```

```
      SUBROUTINE SPLINE (N, X, Y, B, C, D)
      INTEGER N
      DOUBLE PRECISION X(N), Y(N), B(N), C(N), D(N)
C
C  THE COEFFICIENTS B(I), C(I), AND D(I), I=1,2,...,N ARE COMPUTED
C  FOR A CUBIC INTERPOLATING SPLINE
C
C    S(X) = Y(I) + B(I)*(X-X(I)) + C(I)*(X-X(I))**2 + D(I)*(X-X(I))**3
C
C    FOR  X(I) .LE. X .LE. X(I+1)
C
C  INPUT..
C
C    N = THE NUMBER OF DATA POINTS OR KNOTS (N.GE.2)
C    X = THE ABSCISSAS OF THE KNOTS IN STRICTLY INCREASING ORDER
C    Y = THE ORDINATES OF THE KNOTS
C
C  OUTPUT..
C
C    B, C, D  = ARRAYS OF SPLINE COEFFICIENTS AS DEFINED ABOVE.
C
C  USING  P  TO DENOTE DIFFERENTIATION,
C
C    Y(I) = S(X(I))
C    B(I) = SP(X(I))
C    C(I) = SPP(X(I))/2
C    D(I) = SPPP(X(I))/6  (DERIVATIVE FROM THE RIGHT)
C
C  THE ACCOMPANYING FUNCTION SUBPROGRAM  SEVAL  CAN BE USED
C  TO EVALUATE THE SPLINE.
C
C
      INTEGER NM1, IB, I
      DOUBLE PRECISION T
C
      NM1 = N-1
      IF ( N .LT. 2 ) RETURN
      IF ( N .LT. 3 ) GO TO 50
C
C  SET UP TRIDIAGONAL SYSTEM
C
C  B = DIAGONAL, D = OFFDIAGONAL, C = RIGHT HAND SIDE.
C
      D(1) = X(2) - X(1)
      C(2) = (Y(2) - Y(1))/D(1)
      DO 10 I = 2, NM1
         D(I) = X(I+1) - X(I)
         B(I) = 2.*(D(I-1) + D(I))
         C(I+1) = (Y(I+1) - Y(I))/D(I)
         C(I) = C(I+1) - C(I)
```

```
   10 CONTINUE
C
C  END CONDITIONS.  THIRD DERIVATIVES AT  X(1)  AND  X(N)
C  OBTAINED FROM DIVIDED DIFFERENCES
C
      B(1) = -D(1)
      B(N) = -D(N-1)
      C(1) = 0.
      C(N) = 0.
      IF ( N .EQ. 3 ) GO TO 15
      C(1) = C(3)/(X(4)-X(2)) - C(2)/(X(3)-X(1))
      C(N) = C(N-1)/(X(N)-X(N-2)) - C(N-2)/(X(N-1)-X(N-3))
      C(1) = C(1)*D(1)**2/(X(4)-X(1))
      C(N) = -C(N)*D(N-1)**2/(X(N)-X(N-3))
C
C  FORWARD ELIMINATION
C
   15 DO 20 I = 2, N
         T = D(I-1)/B(I-1)
         B(I) = B(I) - T*D(I-1)
         C(I) = C(I) - T*C(I-1)
   20 CONTINUE
C
C  BACK SUBSTITUTION
C
      C(N) = C(N)/B(N)
      DO 30 IB = 1, NM1
         I = N-IB
         C(I) = (C(I) - D(I)*C(I+1))/B(I)
   30 CONTINUE
C
C  C(I) IS NOW THE SIGMA(I) OF THE TEXT
C
C  COMPUTE POLYNOMIAL COEFFICIENTS
C
      B(N) = (Y(N) - Y(NM1))/D(NM1) + D(NM1)*(C(NM1) + 2.*C(N))
      DO 40 I = 1, NM1
         B(I) = (Y(I+1) - Y(I))/D(I) - D(I)*(C(I+1) + 2.*C(I))
         D(I) = (C(I+1) - C(I))/D(I)
         C(I) = 3.*C(I)
   40 CONTINUE
      C(N) = 3.*C(N)
      D(N) = D(N-1)
      RETURN
C
   50 B(1) = (Y(2)-Y(1))/(X(2)-X(1))
      C(1) = 0.
      D(1) = 0.
      B(2) = B(1)
      C(2) = 0.
```

```
D(2) = 0.
RETURN
END
```

```
      DOUBLE PRECISION FUNCTION SEVAL(N, U, X, Y, B, C, D)
      INTEGER N
      DOUBLE PRECISION  U, X(N), Y(N), B(N), C(N), D(N)
C
C  THIS SUBROUTINE EVALUATES THE CUBIC SPLINE FUNCTION
C
C    SEVAL = Y(I) + B(I)*(U-X(I)) + C(I)*(U-X(I))**2 + D(I)*(U-X(I))**3
C
C    WHERE  X(I) .LT. U .LT. X(I+1), USING HORNER'S RULE
C
C  IF  U .LT. X(1) THEN  I = 1  IS USED.
C  IF  U .GE. X(N) THEN  I = N  IS USED.
C
C  INPUT..
C
C    N = THE NUMBER OF DATA POINTS
C    U = THE ABSCISSA AT WHICH THE SPLINE IS TO BE EVALUATED
C    X,Y = THE ARRAYS OF DATA ABSCISSAS AND ORDINATES
C    B,C,D = ARRAYS OF SPLINE COEFFICIENTS COMPUTED BY SPLINE
C
C  IF  U  IS NOT IN THE SAME INTERVAL AS THE PREVIOUS CALL, THEN A
C  BINARY SEARCH IS PERFORMED TO DETERMINE THE PROPER INTERVAL.
C
      INTEGER I, J, K
      DOUBLE PRECISION DX
      DATA I/1/
      IF ( I .GE. N ) I = 1
      IF ( U .LT. X(I) ) GO TO 10
      IF ( U .LE. X(I+1) ) GO TO 30
C
C  BINARY SEARCH
C
   10 I = 1
      J = N+1
   20 K = (I+J)/2
      IF ( U .LT. X(K) ) J = K
      IF ( U .GE. X(K) ) I = K
      IF ( J .GT. I+1 ) GO TO 20
C
C  EVALUATE SPLINE
C
   30 DX = U - X(I)
      SEVAL = Y(I) + DX*(B(I) + DX*(C(I) + DX*D(I)))
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION FMIN(AX,BX,F,TOL)
      DOUBLE PRECISION AX,BX,F,TOL
C
C      AN APPROXIMATION  X  TO THE POINT WHERE  F  ATTAINS A MINIMUM  ON
C  THE INTERVAL  (AX,BX)  IS DETERMINED.
C
C
C
C  INPUT..
C
C  AX    LEFT ENDPOINT OF INITIAL INTERVAL
C  BX    RIGHT ENDPOINT OF INITIAL INTERVAL
C  F     FUNCTION SUBPROGRAM WHICH EVALUATES  F(X)  FOR ANY  X
C        IN THE INTERVAL  (AX,BX)
C  TOL   DESIRED LENGTH OF THE INTERVAL OF UNCERTAINTY OF THE FINAL
C        RESULT ( .GE. 0.0D0)
C
C
C
C  OUTPUT..
C
C  FMIN  ABCISSA APPROXIMATING THE POINT WHERE  F  ATTAINS A MINIMUM
C
C
C      THE METHOD USED IS A COMBINATION OF  GOLDEN  SECTION  SEARCH  AND
C  SUCCESSIVE PARABOLIC INTERPOLATION.  CONVERGENCE IS NEVER MUCH SLOWER
C  THAN  THAT  FOR  A  FIBONACCI SEARCH.  IF  F  HAS A CONTINUOUS SECOND
C  DERIVATIVE WHICH IS POSITIVE AT THE MINIMUM (WHICH IS NOT  AT  AX  OR
C  BX),  THEN  CONVERGENCE  IS  SUPERLINEAR, AND USUALLY OF THE ORDER OF
C  ABOUT  1.324....
C      THE FUNCTION  F  IS NEVER EVALUATED AT TWO POINTS CLOSER TOGETHER
C  THAN  EPS*ABS(FMIN) + (TOL/3), WHERE EPS IS  APPROXIMATELY THE SQUARE
C  ROOT  OF  THE  RELATIVE  MACHINE  PRECISION.   IF   F   IS A UNIMODAL
C  FUNCTION AND THE COMPUTED VALUES OF   F   ARE  ALWAYS  UNIMODAL  WHEN
C  SEPARATED BY AT LEAST  EPS*ABS(X) + (TOL/3), THEN  FMIN  APPROXIMATES
C  THE ABCISSA OF THE GLOBAL MINIMUM OF  F  ON THE INTERVAL  AX,BX  WITH
C  AN ERROR LESS THAN  3*EPS*ABS(FMIN) + TOL.  IF   F   IS NOT UNIMODAL,
C  THEN FMIN MAY APPROXIMATE A LOCAL, BUT PERHAPS NON-GLOBAL, MINIMUM TO
C  THE SAME ACCURACY.
C      THIS FUNCTION SUBPROGRAM IS A SLIGHTLY MODIFIED  VERSION  OF  THE
C  ALGOL  60 PROCEDURE  LOCALMIN  GIVEN IN RICHARD BRENT, ALGORITHMS FOR
C  MINIMIZATION WITHOUT DERIVATIVES, PRENTICE - HALL, INC. (1973).
C
C
      DOUBLE PRECISION  A,B,C,D,E,EPS,XM,P,Q,R,TOL1,TOL2,U,V,W
      DOUBLE PRECISION  FU,FV,FW,FX,X
      DOUBLE PRECISION  DABS,DSQRT,DSIGN
C
C  C IS THE SQUARED INVERSE OF THE GOLDEN RATIO
C
      C = 0.5D0*(3. - DSQRT(5.0D0))
C
```

```
C
C  EPS IS APPROXIMATELY THE SQUARE ROOT OF THE RELATIVE MACHINE
C  PRECISION.
C
      EPS = 1.0D00
   10 EPS = EPS/2.0D00
      TOL1 = 1.0D0 + EPS
      IF (TOL1 .GT. 1.0D00) GO TO 10
      EPS = DSQRT(EPS)
C
C  INITIALIZATION
C
      A = AX
      B = BX
      V = A + C*(B - A)
      W = V
      X = V
      E = 0.0D0
      FX = F(X)
      FV = FX
      FW = FX
C
C  MAIN LOOP STARTS HERE
C
   20 XM = 0.5D0*(A + B)
      TOL1 = EPS*DABS(X)  + TOL/3.0D0
      TOL2 = 2.0D0*TOL1
C
C  CHECK STOPPING CRITERION
C
      IF (DABS(X - XM) .LE. (TOL2 - 0.5D0*(B - A))) GO TO 90
C
C IS GOLDEN-SECTION NECESSARY
C
      IF (DABS(E) .LE. TOL1) GO TO 40
C
C  FIT PARABOLA
C
      R = (X - W)*(FX - FV)
      Q = (X - V)*(FX - FW)
      P = (X - V)*Q - (X - W)*R
      Q = 2.0D00*(Q - R)
      IF (Q .GT. 0.0D0) P = -P
      Q =  DABS(Q)
      R = E
      E = D
C
C  IS PARABOLA ACCEPTABLE
C
   30 IF (DABS(P) .GE. DABS(0.5D0*Q*R)) GO TO 40
      IF (P .LE. Q*(A - X)) GO TO 40
```

```
      IF (P .GE. Q*(B - X)) GO TO 40
:
:  A PARABOLIC INTERPOLATION STEP
:
      D = P/Q
      U = X + D
:
:  F MUST NOT BE EVALUATED TOO CLOSE TO AX OR BX
:
      IF ((U - A) .LT. TOL2) D = DSIGN(TOL1, XM - X)
      IF ((B - U) .LT. TOL2) D = DSIGN(TOL1, XM - X)
      GO TO 50
:
:  A GOLDEN-SECTION STEP
:
   40 IF (X .GE. XM) E = A - X
      IF (X .LT. XM) E = B - X
      D = C*E
:
:  F MUST NOT BE EVALUATED TOO CLOSE TO X
:
   50 IF (DABS(D) .GE. TOL1) U = X + D
      IF (DABS(D) .LT. TOL1) U = X + DSIGN(TOL1, D)
      FU = F(U)
:
:  UPDATE  A, B, V, W, AND X
:
      IF (FU .GT. FX) GO TO 60
      IF (U .GE. X) A = X
      IF (U .LT. X) B = X
      V = W
      FV = FW
      W = X
      FW = FX
      X = U
      FX = FU
      GO TO 20
   60 IF (U .LT. X) A = U
      IF (U .GE. X) B = U
      IF (FU .LE. FW) GO TO 70
      IF (W .EQ. X) GO TO 70
      IF (FU .LE. FV) GO TO 80
      IF (V .EQ. X) GO TO 80
      IF (V .EQ. W) GO TO 80
      GO TO 20
   70 V = W
      FV = FW
      W = U
      FW = FU
      GO TO 20
```

```
80 V = U
   FV = FU
   GO TO 20
C
C  END OF MAIN LOOP
C
90 FMIN = X
   RETURN
   END
```

```
*******************************************************************************
                                                                              *
 PROGRAM NAME : MAPGRAF                                                       *
 WRITTEN BY : J.F. BRANDEAU                                                   *
 COMPILER(S) : WATFIV  (DOUBLE PRECISION)                                     *
           --------------------------------------------------                 *
 PURPOSE : USE COEFFICIENTS X TO PRODUCE DATA POINTS THAT WILL BE READ*
 BY SAS TO PRODUCE A 3-D PLOT OF A STRESS-STRAIN SURFACE.  PROGRAM    *
 CREATES A GRID OF STRAIN VS. LOG STRAIN RATE, AND USES SUBROUTINE    *
 ZEROIN TO SOLVE FOR THE VALUE OF STRESS FOR EACH UNIQUE COMBINATION. *
 THE DENSITY OF THE GRID, THE RANGE OF THE GRID, AND THE NON-         *
 PERMISSIBLE REGION ARE ALL CONTROLLABLE.  THE NON-PERMISSIBLE REGION *
 OF THE GRID WILL HAVE NO POINTS ON IT, TO PREVENT EXTRAPOLATION PAST *
 EXPERIMENTAL LIMITS.  THE SHAPE OF THE EDGE IS CALCULATED USING A    *
 SPLINE CURVE FIT TO THE MAX VALUES OF STRAIN FOR EACH OBSERVED STRESS*
 THIS IS DONE BY SUBROUTINE SPLINE AND SEVAL.                         *
           --------------------------------------------------         *
 VARIABLES :                                                          *
                                                                      *
 A1 & B1 : A1 IS THE MAX OBSERVED VALUES OF STRAIN, ONE FOR EACH VALUE*
  OF STRAIN RATE B1.   B1 MUST START WITH LOWEST VALUE OF STRAIN RATE *
  FIRST, OR THE SPLINE WILL BE INCORRECTLY CALCULATED.                *
                                                                      *
 K : THE NUMBER OF UNIQUE STRAIN RATES OBSERVED, ALSO THE NUMBER OF   *
  ELEMENTS I EACH OF A1 & B1.                                         *
                                                                      *
 SIGMAX : GREATER THAN THE EXPECTED MAX VALUE OF STRESS (KSI) TO BE   *
  USED AS THE UPPER LIMIT OF SEARCH FOR ZEROIN.  FOR EACH UNIQUE      *
  STRAIN RATE THIS MAY BE CORRECTED DOWNWARD BY THE PROGRAM TO PREVENT*
  OVERFLOWS.  THIS IS SET BY THE USER.                                *
                                                                      *
 A,B,N,D,C & COEFF : A,B,N,D, & C ARE THE MANTISSA OF THE COEFFICIENTS*
  FOR THE EQUATION, AND COEFF IS THE EXPONENTS.  THE PRODUCTS         *
  A * COEFF(1), B * COEFF(2), ETC... ARE THE COEFFICIENTS.            *
                                                                      *
 X1 : THE STARTING POINT FOR VALUES OF STRAIN.  THIS MUST BE GREATER  *
  THAN ZERO, AS THE EQUATION IS INDETERMINATE @ STRAIN = 0.0.         *
                                                                      *
 Y1 : THE STARTING POINT FOR LOG RATE.  Y1 = -3 IS A STARTING POINT OF*
  STRAIN RATE = 0.001 / SEC.                                          *
                                                                      *
 DY : STEP LENGTH IN LOG RATE (OR Y) DIRECTION.                       *
                                                                      *
 DX :  "     "     "  STRAIN (OR X)        "                          *
                                                                      *
 YMAX : MAX VALUE OF LOG RATE TO BE USED & IS ASSIGNED BY THE USER.   *
                                                                      *
 THOLD = MAX VALUE OF STRAIN TO USE, & IS SET BY PROGRAM TO EQUAL     *
  MAX VALUE IN A1.                                                    *
                                                                      *
```

```
: TOL : CONVERGENCE CRITERIA FOR ZEROIN.                                *
:                                                                       *
: IX : # OF POINTS ON STRAIN (X) AXIS.                                  *
:                                                                       *
: IY : "  "  "      "  LOG RATE (Y) "                                   *
:     THE TOTAL GRID WILL CONSIST OF IY * IX - NON-PERMISSIBLE POINTS.  *
:                                                                       *
: C1,D1,E1 : WORK VECTORS FOR SPLINE & SEVAL.  MUST NOT BE CHANGED.     *
:          --------------------------------------------------           *
: I/O REQUIREMENTS :                                                    *
:                                                                       *
: FILE #5 : MANTISSA OF COEFFICIENT VECTOR AND EXPONENT OF COEFFICIENT  *
:          VECTOR ON TWO RECORDS.                                       *
:          --------------------------------------------------           *
: OPTIONS :       NONE                                                  *
:                                                                       *
:************************************************************************
      IMPLICIT REAL * 8 (A-H, N, O-Z)
      EXTERNAL FUNCT
      DIMENSION Y(40), X(40), YE(40), COEFF(5)
      DIMENSION A1(6), B1(6), C1(6), D1(6), E1(6)
      COMMON X, Y, I, J, A, B, C, D, N, RATEB, RATED, TLOG
      COMMON /SUB1/ GEPS
      DATA A1/6.90D-3, 6.5D-3, 5.7D-3, 5.1D-3,    10.5D-3, 9.2D-3/
      DATA B1 /.001, 0.10, 10.0, 150.0, 0.0, 0.0/
:
:  SET PROGRAM PARAMETERS
:
      K = 4
      YMAX = 2.3
      SIGMAX = 25.0D0
      X1 = 0.05D-3  ;   Y1 = -3.0D0
      TOL = 1.0D-8  ;   IX = 30    ;   IY = 30
      THOLD = 0.0
       DO 5 J = 1, K
       IF (A1(J) .GT. THOLD) THOLD = A1(J)
    5  B1(J) = DLOG10 (B1(J))
:
      CALL SPLINE (K, B1, A1, C1, D1, E1)
      READ (5,*) A, B, N, D, C
      READ (5,*) COEFF
      A = A * COEFF(1)   ;  B = B * COEFF(2) ;  N = N * COEFF(3)
      D = D * COEFF(4)   ;  C = C * COEFF(5)
      PRINT, 'A  ',A   ;   PRINT, 'B  ',B  ;   PRINT, 'N ',N
      PRINT, 'D  ',D   ;   PRINT,'C ',C
:
:  CHECK FOR VALUES THAT WILL CAUSE OVER/UNDER FLOW
:
      IF (A .EQ. 0.0D0) THEN DO
       TLOG = -80.0D0
```

```
      ELSE DO
       TLOG = DLOG10 (A)
      ENDIF
C
      DX = (THOLD - X1) / DFLOAT(IX-1)
      DY = (YMAX - Y1) / DFLOAT(IY-1)
          DO 10 J = 1, IY
          Y(J) = DFLOAT(J-1) * DY + Y1
 10       YE(J) = 10.0 ** Y(J)
          DO 20 I = 1, IX
 20       X(I) = DFLOAT(I-1) * DX + X1
C
C    CALCULATE MACHINE EPSILON FOR ZEROIN
C
      GEPS = 1.0D0
 24   GEPS = GEPS/2.0D0
      TOL1 = 1.0D0 + GEPS
      IF (TOL1 .GT. 1.0D0) GO TO 24
C
C    BEGIN SOLUTIONS
C
      DO 40 J = 1, IY
C
C    CORRECT BX IF NEEDED TO PREVENT OVERFLOW
C
      BX = SIGMAX
      RATEB = DLOG10 (YE(J) ** B)
 25   AX = N * DLOG10 (BX) + RATEB
       IF (AX .GT. 75.0D0) THEN DO
        BX = BX - 0.5D0
        GO TO 25
       ENDIF
C
      RATEB = YE(J) ** B
      RATED = C * YE(J) ** D
      Z =  X(1) * RATED / 1.3
      TEMP = SEVAL (K, Y(J), B1, A1, C1, D1, E1)
C
C    ALLOW FOR SPLINE RIPPLE
      IF (TEMP .GT. (THOLD * 0.97D0) ) TEMP = THOLD
C
        DO 30 I = 1, IX
        IF (X(I) .GT. TEMP) GO TO 40
        AX = Z
        Z = ZEROIN (AX, BX, FUNCT, TOL)
        WRITE (6,400) X(I), Y(J), YE(J), Z
 30     CONTINUE
 40   CONTINUE
      STOP
400   FORMAT (1H ,5(1PD13.5))
```

```
      END
      DOUBLE PRECISION FUNCTION FUNCT(STRESS)
      IMPLICIT REAL * 8 (A-H, N, O-Z)
      DIMENSION Y(40), X(40)
      COMMON X, Y, I, J, A, B, C, D, N, RATEB, RATED, TLOG
      T1 = STRESS / RATED
      IF (STRESS .GT. 1.0D-1) GO TO 10
      IF ((N * DLOG10(STRESS)) .GT. -60.0D0) GO TO 10
       HOLD = -25.0D0
       GO TO 15
   10 T2 = STRESS ** N
      HOLD = DLOG10 (T2)
   15 IF ((HOLD + TLOG) .LT. -17.0D0) THEN DO
       FUNCT = T1 - X(I)
      ELSE DO
       FUNCT = T1 + A * T2 * RATEB - X(I)
      ENDIF
   25 RETURN
      END
C$OPTIONS NOLIST
      DOUBLE PRECISION FUNCTION ZEROIN(AX,BX,F,TOL)
      DOUBLE PRECISION AX,BX,F,TOL
      DOUBLE PRECISION  A,B,C,D,E,EPS,FA,FB,FC,TOL1,XM,P,Q,R,S
      DOUBLE PRECISION  DABS,DSIGN
      COMMON /SUB1/ EPS
      A = AX
      B = BX
      FA = F(A)
      FB = F(B)
   20 C = A
      FC = FA
      D = B - A
      E = D
   30 IF (DABS(FC) .GE. DABS(FB)) GO TO 40
      A = B
      B = C
      C = A
      FA = FB
      FB = FC
      FC = FA
   40 TOL1 = 2.0D0*EPS*DABS(B) + 0.5D0*TOL
      XM = .5*(C - B)
      IF (DABS(XM) .LE. TOL1) GO TO 90
      IF (FB .EQ. 0.0D0) GO TO 90
      IF (DABS(E) .LT. TOL1) GO TO 70
      IF (DABS(FA) .LE. DABS(FB)) GO TO 70
      IF (A .NE. C) GO TO 50
      S = FB/FA
      P = 2.0D0*XM*S
      Q = 1.0D0 - S
```

```
      GO TO 60
   50 Q = FA/FC
      R = FB/FC
      S = FB/FA
      P = S*(2.0D0*XM*Q*(Q - R) - (B - A)*(R - 1.0D0))
      Q = (Q - 1.0D0)*(R - 1.0D0)*(S - 1.0D0)
   60 IF (P .GT. 0.0D0) Q = -Q
      P = DABS(P)
      IF ((2.0D0*P) .GE. (3.0D0*XM*Q - DABS(TOL1*Q))) GO TO 70
      IF (P .GE. DABS(0.5D0*E*Q)) GO TO 70
      E = D
      D = P/Q
      GO TO 80
   70 D = XM
      E = D
   80 A = B
      FA = FB
      IF (DABS(D) .GT. TOL1) B = B + D
      IF (DABS(D) .LE. TOL1) B = B + DSIGN(TOL1, XM)
      FB = F(B)
      IF ((FB*(FC/DABS(FC))) .GT. 0.0D0) GO TO 20
      GO TO 30
   90 ZEROIN = B
      RETURN
      END
      SUBROUTINE SPLINE (N, X, Y, B, C, D)
      INTEGER N
      DOUBLE PRECISION X(N), Y(N), B(N), C(N), D(N)
      INTEGER NM1, IB, I
      DOUBLE PRECISION T
      NM1 = N-1
      IF ( N .LT. 2 ) RETURN
      IF ( N .LT. 3 ) GO TO 50
      D(1) = X(2) - X(1)
      C(2) = (Y(2) - Y(1))/D(1)
      DO 10 I = 2, NM1
         D(I) = X(I+1) - X(I)
         B(I) = 2.*(D(I-1) + D(I))
         C(I+1) = (Y(I+1) - Y(I))/D(I)
         C(I) = C(I+1) - C(I)
   10 CONTINUE
      B(1) = -D(1)
      B(N) = -D(N-1)
      C(1) = 0.
      C(N) = 0.
      IF ( N .EQ. 3 ) GO TO 15
      C(1) = C(3)/(X(4)-X(2)) - C(2)/(X(3)-X(1))
      C(N) = C(N-1)/(X(N)-X(N-2)) - C(N-2)/(X(N-1)-X(N-3))
      C(1) = C(1)*D(1)**2/(X(4)-X(1))
      C(N) = -C(N)*D(N-1)**2/(X(N)-X(N-3))
```

```
15 DO 20 I = 2, N
      T = D(I-1)/B(I-1)
      B(I) = B(I) - T*D(I-1)
      C(I) = C(I) - T*C(I-1)
20 CONTINUE
   C(N) = C(N)/B(N)
   DO 30 IB = 1, NM1
      I = N-IB
      C(I) = (C(I) - D(I)*C(I+1))/B(I)
30 CONTINUE
   B(N) = (Y(N) - Y(NM1))/D(NM1) + D(NM1)*(C(NM1) + 2.*C(N))
   DO 40 I = 1, NM1
      B(I) = (Y(I+1) - Y(I))/D(I) - D(I)*(C(I+1) + 2.*C(I))
      D(I) = (C(I+1) - C(I))/D(I)
      C(I) = 3.*C(I)
40 CONTINUE
   C(N) = 3.*C(N)
   D(N) = D(N-1)
   RETURN
50 B(1) = (Y(2)-Y(1))/(X(2)-X(1))
   C(1) = 0.
   D(1) = 0.
   B(2) = B(1)
   C(2) = 0.
   D(2) = 0.
   RETURN
   END
   DOUBLE PRECISION FUNCTION SEVAL(N, U, X, Y, B, C, D)
   INTEGER N
   DOUBLE PRECISION  U, X(N), Y(N), B(N), C(N), D(N)
   INTEGER I, J, K
   DOUBLE PRECISION DX
   DATA I/1/
   IF ( I .GE. N ) I = 1
   IF ( U .LT. X(I) ) GO TO 10
   IF ( U .LE. X(I+1) ) GO TO 30
10 I = 1
   J = N+1
20 K = (I+J)/2
   IF ( U .LT. X(K) ) J = K
   IF ( U .GE. X(K) ) I = K
   IF ( J .GT. I+1 ) GO TO 20
30 DX = U - X(I)
   SEVAL = Y(I) + DX*(B(I) + DX*(C(I) + DX*D(I)))
   RETURN
   END
```

```
C$OPTIONS DEC,CCOMP=19
C  PROGRAM NAME: STRESS
      REAL DUMMY(24), OUT(6), X(15), L95(4), YOI(15)
      REAL ROJ, AREA(15), DLT, SUBJL, SUBJW, L1L2, W1W2, DELX(15)
      REAL INER1, INER2, MASS, SIG(2), FORCE(6,4), MOMENT(3)
      REAL A, B, C, CG(15), WEIGHT(15), INERT(45), DI(4)
      REAL OMEGA(3), FJ(3), TJ(3), ALPHA(3), ACCEL(3), U2(3)
      REAL POSIT(24), CNSTRN(6)
      INTEGER SEG(8), CHECK(4), NCON, KCON, YES, KT6, J, K, NSEGS, NT, I
      INTEGER ODD
      EQUIVALENCE (DUMMY(1), FORCE(1)), (T, TEMP), (S2, TEMP, SUBJL)
      EQUIVALENCE (A1, S1, SUBJW), (INER2, SUBJL)
      COMMON /Q5/ FORCE
      COMMON /Q6/ BONEP(8), BONEA(8), BONEI(8), L1L2, W1W2, A1, INER1,
     $       RORIG, INER2, RO
      DATA L95 /18.89, 16.98, 11.02, 11.0 /
      DATA SEG /'RUL','FLL','LUL','LLL','RUA','RLA','LUA','LLA' /
      DATA GINCH / 386.0886 /, PI / 3.141593 /, W95 / 217.0 /
      DET (A,B) = SQRT (A * A + B * B / 2. )
      S1MAX = 0.0
      S2MAX = 0.0
      TMAX = 0.0
      WRITE (8,5)
    5 FORMAT (' INPUT SEGMENT # TO BE ANALYZED (INTEGER) ??????'/)
      DO 7 I = 1, 8
        WRITE (8,6) I, SEG(I)
    6 FORMAT (1H ,I3,' = ',A4)
    7 CONTINUE
      READ (1,*) J
      NSEGS = 7
      READ (5) NT, DLT, KCON4, (POSIT(I), I = 1, KCON4)
      YES = 0
      DO 8 I = 1, KCON4, 4
        IF (POSIT(I) .NE. J) GO TO 8
        YES = I
        GO TO 9
    8 CONTINUE
C-----------------------------------------------------------------------
C  FIND MASS, DI AND SEMI-MAJOR AXES OF SEGMENT J FROM FILE #5
C-----------------------------------------------------------------------
    9 READ (5) DI
      IF (J .NE. 1) CALL FOR (J-1, 5)
      READ (5) K, MASS, A, B, C
      IF (J .NE. 8) CALL FOR (8-J, 5)
      IF (K .NE. J) PRINT,'LOCKING FOR SEGMENT FILE',J,' FOUND',K

      WRITE (3,150) J, SEG(J), NT, NSEGS
      WRITE (3,155) MASS, A, B, C
      IL = YES + 1
      IR = YES + 3
```

AD-A112 458   DUKE UNIV DURHAM NC DEPT OF MECHANICAL ENGINEERING A--ETC F/G 6/19
DEVELOPMENT OF A STRAIN RATE DEPENDENT LONG BONE INJURY CRITERI--ETC(U)
JAN 82 T K HIGHT   AFOSR-81-0062

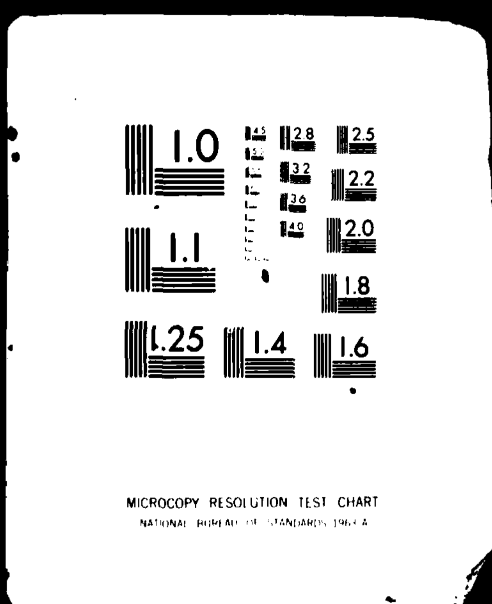
UNCLASSIFIED   AFOSR-TR-82-0138   NL

2 of 2

END
DATE FILMED
4-82
DTIC

2 OF 2

AD
A112458



```
      IF (YES .GT. 0) WRITE (3,156) (POSIT(I), I = IL, IR)
C-----------------------------------------------------------------------
C   SET JNUM TO EQUAL PROPER COUNTER IN GEOMETRY FILE #5.
C-----------------------------------------------------------------------
      READ (6,*) CHECK
      GO TO (10, 20, 10, 20, 30, 40, 30, 40), J
   10 JNUM = 1
        ODD = 1
        GO TO 50
   20 JNUM = 2
        ODD = -1
        CALL FOR (CHECK(1), 6)
        GO TO 50
   30 JNUM = 3
        ODD = 1
        CALL FOR (CHECK(1) + CHECK(2), 6)
        GO TO 50
   40 JNUM = 4
        ODD = -1
        CALL FOR (CHECK(1) + CHECK(2) + CHECK(3), 6)
   50 WRITE (3,157) DI(JNUM), L95(JNUM)
C-----------------------------------------------------------------------
C   CREATE PERCENT-OF-LENGTH POINTS TO BE USED BETWEEN 20 & 80
C   SOLVE FOR NEEDED VALUES @ EACH FREE-BODY SECTION.
C-----------------------------------------------------------------------
      VOL = 4. * PI * A * B * B / 3.
      RO = MASS / VOL
      A2 = A * A
      A3 = A * A2
      A4 = A * A3
      B2 = B * B
      B3 = B * B2
      WRITE (3,425)
      DO 60 I = 1, NSEGS
        K = 3 * I - 2
        DELX(I) = 0.20 + FLOAT (I-1) * 0.60 / FLOAT (NSEGS-1)
        X(I) = DELX(I) * L95(JNUM)
C
C   XI WITH RESPECT TO C.G. OF FULL ELLIPSOID
C
        X1 = X(I) - DI(JNUM)
        X2 = X1 * X1
        X3 = X2 * X1
        X4 = X3 * X1
C
C   FIND WEIGHT (OR MASS) OF FREE-BODY SECTION I.
C
        WEIGHT(I) = RO * PI * B2 * (A2*X1 - X3/3. + 2.*A3/3.) / A2
C   FIND C.G. OF FREE-BODY SECTION I RELATIVE TO C.G. OF ELLIPSOID
C
```

```
      CG(I) = 3. * (2.*X2*A2 - X4 - A4) / (3.*A2*X1 - X3 + 2.*A3) / 4.
C
C  FIND I(XX) OF FREE-BODY SECTION I ABOUT POINT X(I)
C
      TEMP = PI * RO * B2 * B2 / (2. * A4) *
     $   (A4*X1 + X4*X1/5. - 2.*A2*X3/3. + 8.*A2*A3/15.)
      INERT(K) = TEMP / GINCH
C
C  FIND I(YY) = I(ZZ) OF FREE-BODY SECTION I ABOUT POINT X(I)
C
      TEMP = RO * PI * B2 * (X3* (1./3. - X2 / (30.*A2))
     $ +  X1*B2/4. * (1. - 2.*X2/(3.*A2) + X4/(5.*A4))
     $     + 2. / 15. * (A3 + A*B2) + A2*X1/2. + 2.*X2*A/3. )
      INERT(K+1) = TEMP / GINCH
      INERT(K+2) = INERT(K+1)
      WRITE (3,450) I, XI, WEIGHT(I), CG(I), INERT(K), INERT(K+1)
C
   60 CONTINUE
C-----------------------------------------------------------------------
C  FIND AREAS AND INERTIA PROPERTIES -- USE DETAILED DATA IF
C  AVAILABLE (CHECK(JNUM) = +1) -- ELSE USE CONSTANT CROSS-SECTION
C  HOLLOW TUBE FOR BONE (CHECK(JNUM) = -1).
C-----------------------------------------------------------------------
      IF (CHECK(JNUM) .GT. 0) GO TO 80
C
C  CONSTANT CROSS-SECTION
C
      READ (6,*) A1, TEMP4, TEMP1
      TEMP2 = TEMP4 * TEMP4
      TEMP3 = TEMP1 * TEMP1
      A1 = PI * (TEMP3 - TEMP2)
      TEMP = PI * (TEMP3 * TEMP3 - TEMP2 * TEMP2) / 4.
      WRITE (3,160) TEMP4, TEMP1, A1, TEMP
      TEMP = TEMP1 / TEMP
      DO 70 I = 1, NSEGS
        AREA(I) = A1
   70 YOI(I) = TEMP
      GO TO 95
C
C  VARIABLE CROSS-SECTION WITH SCALING OF 95TH PERCENTILE
C
   80 READ (6,*) SUBJL, SUBJW
      K = CHECK(JNUM) - 1
      DO 85 I = 1, K
   85 READ (6,*) BONEP(I), BONEA(I), BONEI(I)
      K = 1
      LIL2 = SUBJL / L95(JNUM)
      WIW2 = SUBJW / W95
      WRITE (3,170) SUBJW, SUBJL, W95, L95(JNUM)
      WRITE (3,175)
```

```
      DO 90 I = 1, NSEGS
        CALL INTERP (K, JNUM, DELX(I), AREA(I), YOI(I))
        WRITE (3,180) DELX(I), X(I), AREA(I), INER2, RO
C4      WRITE (4,400) DELX(I),X(I),AI,INERI,RORIG,AREA(I),INER2,RO
   90 CONTINUE
C4    STOP
C-----------------------------------------------------------------------
C   BEGIN MAIN LOOP FOR ALL TIME STEPS FOR LIMB # J
C-----------------------------------------------------------------------
   95 DO I45 KK = I, NT
C
C   FIND APPLIED LOADS ON LIMB # J FROM FILE #5.
C
        READ (5) TIME
        IF (J .NE. 1) CALL FOR (J-1, 5)
        READ (5) LIMB, KT, NCON, OMEGZ, FI, TI, ALPHA, ACCEL, OZ,
     $   (DUMMY(I), I = 1, KT), (CNSTRN(I), I = 1, NCON)
        IF (J .NE. 8) CALL FOR (8-J, 5)
        WRITE (3,200) TIME
CI      WRITE (3,2) FI, TI, ALPHA, ACCEL, OMEGA, OZ
C1    2 FORMAT (1H ,'APPLIED LOADS',6F12.5,/,(T15,6F12.5))
      KT6 = KT / 6
      IF (KT6 .EQ. 0) GO TO 96
      WRITE (3,500) KT6, (DUMMY(L), L = 1, KT, 6)
C1    PRINT,(DUMMY(L), L = 1, KT)
C1 96 IF (YES .LE. 0) GO TO 100
C1    PRINT,'"OTHER CONSTRAINT FORCES"'
C1    PRINT,(POSIT(L), L = YES, YES), CNSTRN(I), I = I, 3)
C-----------------------------------------------------------------------
C   BEGIN LOOP FOR EACH FREE-BODY SECTION I.
C   CALCULATE STRESSES ON EACH FREE-BODY SECTION USING MOHR'S CIRCLE
C   AND STATIC EQUILIBRIUM.
C-----------------------------------------------------------------------
  100 WRITE (3,250)
      DO 145 I = 1, NSEGS
      ROI = YOI(I) / 2
      XFORCE = 0.0
        DO I05 K = 1, 3
  105   MOMENT(K) = 0.0
      IF (YES .LE. 0) GO TO 107
C
C   FIND MOMENTS AND SUM FORCES DUE TO "OTHER CONSTRAINT FORCES"
C
      XPOINT = X(I) - POSIT(YES+1)
      IF (XPOINT) 107, 107, 106
  106 MOMENT(1) = POSIT(YES+2) * CNSTRN(3) - POSIT(YES+3) *        N(2)
      MOMENT(2) = XPOINT * CNSTRN(3) + POSIT(YES+3) * CNSTRN(1)
      MOMENT(3) = -XPOINT * CNSTRN(2) - POSIT(YES+2) * CNSTRN(1)
      XFORCE = CNSTRN(1)
  107 IF (KT .EQ. I) GO TO 120
```

```
  FIND MOMENTS & FORCES DUE TO CONTACTS

      DO 110 K = 1, K16
        XPOINT = X(I) - FORCE(1,K)
        IF (XPOINT) 120, 120, 108
108     XFORCE = XFORCE + FORCE(4,K)
110     CALL XPROD (XPOINT, K, MOMENT)
-----------------------------------------------------------------
  FIND TOTAL MOMENT, SHEAR STRESS & TENSILE STRESSES
-----------------------------------------------------------------
  X2 IS DISTANCE FROM X(I) TO C.G. OF SECTION (* = DIST(I) * MASS

120 X2 = (DI(JNUM) - X(I) + CG(I)) * WEIGHT(I)
      K = 3 * I - 2
2     IF (K .GT. 1) WRITE (3,121) MOMENT, XFORCE
2121  FORMAT (' CONTACT MOMENTS',T25,3F10.3,' SUM XFORCE',F10.3)

  MOMENTS ABOUT Z-AXIS

      TEMP1 = INERT(K+2) * (U2(3) - ALPHA(3))  -  MOMENT(3)
     $         -FJ(2) * X(I) -  ACCEL(2) * X2

  MOMENT ABOUT Y AXIS

      TEMP2 = INERT(K+1) * (U2(2) - ALPHA(2))  -  MOMENT(2)
     $         +FJ(3) * X(I)  +  X2 * ACCEL(3)
      IF (ODD .LT. 0) TEMP2 = TEMP2 - TJ(2)

  MOMENT ABOUT X-AXIS (TORSION TORQUE)

      TORQUE = INERT(K) * (U2(1) - ALPHA(1))  -  MOMENT(1)
      IF (ODD .LT. 0) TORQUE = TORQUE - TJ(1)

      T = SQRT (TEMP1 * TEMP1 + TEMP2 * TEMP2)
C2    WRITE (3,123) TEMP1, TEMP2, TEMP1, TEMP2, T
C2123 FORMAT (' MOMENTS (Z,Y,TOTAL)',T25,3F10.3)
      TEMP1 = T * YO1(I)
C
C TOTAL AXIAL FORCE
C
      TEMP2 = ((WEIGHT(I) * ACCEL(1)) - XFORCE - FJ(1)) / AREA(I)
      SIG(1) = TEMP2 + TEMP1
      SIG(2) = TEMP2 - TEMP1
C2    WRITE (3,122) TEMP1, TEMP2, SIG
C2122 FORMAT (' BEND STRESS,FX,S1,S2',T25,4F10.3)
      TAU = TORQUE * RCJ
C2    WRITE (3,124) TORQUE, TAU
C2124 FORMAT (' TORQUE, SHEAR STRESS',T25,2F10.3)
C
```

```
C   SOLVE MOHR'S CIRCLE FOR EACH END OF DIAMETER OF TUBE
C
      DO 125 K2 = 1, 2
         K = K2 * K2
         TEMP = DET (TAU, SIG(K2))
         TEMP2 = SIG(K2) / 2.
         SA = TEMP2 + TEMP
         OUT(K) = SA
         SB = TEMP2 - TEMP
         OUT(K+1) = SB
         TI = TEMP
         OUT(K+2) = TI
  125 CONTINUE
C
      S1 = AMAX1 (OUT(1), OUT(4))
      S2 = AMIN1 (OUT(2), OUT(5))
      T  = AMAX1 (OUT(3), OUT(6))
      IF (S1 .LT. S1MAX) GO TO 130
         S1MAX = S1
         XS1 = X(I)
  130 IF (S2 .GT. S2MAX) GO TO 135
         S2MAX = S2
         XS2 = X(I)
  135 IF (T .LT. TMAX) GO TO 140
         TMAX = T
         XT = X(I)
  140 WRITE (3,300) DELX(I), X(I), OUT
C9    WRITE (7,400) TIME, DELX(I), X(I), OUT
  145 CONTINUE
      STOP
C
  150 FORMAT (1H1,'STRESS HISTORY FOR SEGMENT #',I2,' ',A3,/,
     $   ' THERE ARE',I3,' TIME STEPS'/,
     $   I3,' FREE-BODY SECTIONS WILL BE USED')
  155 FORMAT (1H-,'SEGMENT MASS =',F10.3,' LBS.'/' SEMI-MAJOR AXES',
     $   ' (XYZ) ARE :',3F10.3)
  157 FORMAT ('-PROXIMAL JOINT TO C.G. DISTANCE =',F10.2,' IN.'/
     $   ' PROXIMAL JOINT TO DISTAL JOINT LENGTH =',F10.2,' IN.')
  156 FORMAT ('-"OTHER CONSTRAINT FORCE" LOCATIONS',3F10.3)
     $   ' PROXIMAL JOINT TO DISTAL JOINT LENGTH =',F10.2,' IN.')
  160 FORMAT ('-BONE GEOMETRY -- CONSTANT CROSS-SECTION HOLLOW TUBE'/,
     $ '               INSIDE RADIUS =',F6.3,' IN.'/,
     $ '              OUTSIDE RADIUS =',F6.3,' IN.'/,
     $ '                        AREA =',F6.3,' SQ. IN.'/,
     $ '     2ND MOMENT OF INERTIA =',F6.3,' IN. ** 4'/)
  170 FORMAT ('-GEOMETRY DATA FOR THIS LIMB -- VARIABLE CROSS-SECTION',
     $  ' HOLLOW TUBE'/'          BASED ON EXPERIMENTAL DATA:'/
     $  '          SUBJECT :         WEIGHT =',F7.0,' LBS.     LIMB LENGTH =',
     $  F7.2,' IN.'/' 95TH PERC. MAN :       WEIGHT =',F7.0,' LBS. '
     $  '     LIMB LENGTH =',F7.2,' IN.'/)
```

```
175 FORMAT (1H0,T5,'% L',T13,'X(IN)',T26,'A(IN**2)',T39,
   $  'I(IN**4)',T51,'OUTSIDE R.(IN)'/)
180 FORMAT (1H ,F7.3,4(F9.3,5X))
200 FORMAT ('-TIME = ',F7.3,' SECONDS')
250 FORMAT (1H0,T5,'% L',T13,'X(IN)',T29,'S1T',T39,'S2T',T48,'TAU T',
   $  T64,'S1B',T73,'S2B',T82,'TAU B'/)
300 FORMAT (1H , F7.3, F9.3, 5X, 3F10.2, 5X, 3F10.2)
400 FORMAT (1H ,9F16.5)
425 FORMAT (1H-,T4,'#',T12,'X (FROM C.G.)',T29,'MASS (LB)',T41,
   $  'CG (FROM C.G.)',T62,'I(XX)',T72,'I(YY) = I(ZZ)'/)
450 FORMAT (1H ,I3,2X,5F15.4)
500 FORMAT (1H ,I1,' CONTACT(S) THIS TIME STEP',
   $  ' OCCUR AT X (FROM PROXIMAL) =',4F6.2,/)
    END
C.
    SUBROUTINE XPROD (A, X, K)
    REAL A(3), B(6,4)
    COMMON /Q5/ B
C
    A(1) = A(1) + X * B(2,K) * B(6,K) - B(3,K) * B(5,K)
C
    A(2) = A(2) + X * B(6,K) + B(4,K) * B(3,K)
C
    A(3) = A(3) - X * B(5,K) - B(4,K) * B(2,K)
C
    RETURN
    END
C.
    SUBROUTINE INTERP (K, J, X, A2, YOI)
    REAL L1L2, I1, I2
    COMMON /Q6/ BP(8), BA(8), BI(8), L1L2, W1W2, A1, I1, RORIG, I2,
   $  ROUT
    DATA PI /3.1415926/
C
C  FIND BOUNDING INTERVAL
C
    DO 10 I = K, 6
       IF (BP(I) .GT. X) GO TO 20
 10 CONTINUE
 20 IMI = I - 1
C
C  INTERPOLATE FOR VALUES OF AREA AND INERTIA
C
    PERC = (X - BP(IMI)) / (BP(I) - BP(IMI))
    A1 = BA(IMI) + PERC * (BA(I) - BA(IMI))
    I1 = BI(IMI) + PERC * (BI(I) - BI(IMI))
C3  PRINT,'A1, I1',A1,I1
    K = IMI
C
C  FIND ORIGINAL OUTSIDE RADIUS FOR HOLLOW TUBE
```

```
 SQRT (2. * (AI / 4. + PI * I1 / A1) / PI)
ORIGINAL OUTSIDE RADIUS',RORIG

 TO 95-TH PEPCENTILE SIZE

1W2 * L1L2 * ROPIG / I1 * PI / 4.)
 / W1W2 / PI
 / (X1 * Y1)
X1, Y1, Z1',X1,Y1,Z1
(Z1 + SQFT(Z1 * Z1 + 8. * Y1)) / 4.
NEW OUTSIDE RADIUS',ROUT
UT * ROUT
QRT (SO - YI)
N * RIN
 * (SO - XI)
 * (SO * SO - XI * XI) / 4.
OUT / I2
A2,I2,YOI',A2,I2,YOI



INE FOR (K, IFILE)
LE .EQ. 8) GO TO 20
 = 1, K
FILE)

 = 1, K
FILE, *)

```

ILMED

8